Pi CONNECT
The all-new official Pi remote desktop tool

TOP TERMINAL
Power up your command-line life!

GEFORCE NOW!
Streaming games on the Steam Deck

LINUX FORMAT

The #1 open source mag

EASY INSTALLS
Master all your apps and packages in Ubuntu

REPAIR IT WITH LINUX

Refurbish, fix and reinstall your old systems, and give them a new lease of life!

SAVE YOUR OLD PC!

PLUS: HOW TO

- Create a turn-based internet game
- Use Fyne to design slick interfaces
- Build a Raspberry Pi fortune teller

CLASSIC TERMINAL
Relive your VAX mini days on the DEC VT100

ADMIN MONITORS
Discover Coroot next-gen server monitoring

INSIDE LINUX
Get to grips with the Linux virtual systems

LXF August 2024
FUTURE

LINUX FORMAT

MEET THE TEAM

This issue, we're breathing new life into old PCs, so what's the oldest system you still have running in your lab?

Jonni Bidwell
I have a slide rule here, does that count? With roughly the same computational power, I also have the diet edition of the original Eee PC 700 from 2007. It's been running Arch (now Arch 32) for 16 years. I also have a Sansa Clip MP3 player running *Rockbox* (and *Doom).*

Michael Reed
It's amazing what you can do with an old computer and Linux. A computer that's being thrown out by others can carry out most tasks. The bottleneck tends to be browser tabs, and there's not much you can do about that. Unless my colleagues know some tricks...

Nate Drake
My mother was delighted to learn I'm still using the Acorn Archimedes (3010) that she bought me in 1993. I first used it to reprogram Basic games. These days, I find the lack of internet connection helpful for creative writing. Mum says at least we're getting our money's worth out of it.

Les Pounder
My oldest systems are roughly a decade old. I used to have older machines (X61 Thinkpad) but now my oldest machines are an original Raspberry Pi with 256Mb of RAM and my trusty Lenovo X220 laptop. Hang on, I do have an AMD Athlon 1GHz in the loft somewhere.

Matthew Holder
My oldest running PC is my incredibly low-power (in terms of compute) HP Micro Server. Over the years, it's had a fan and the PSU changed, a network card due to lightning frying the on-board one, and multiple hard drives. Let's see if it's still going strong in another 13 years!

*Savings are based on the cover price.

Rescue and repair

We often like to have a little giggle with our covers: "What's that crazy Tux up to this month?" This issue, though, the subject of ewaste isn't really a laughing matter. It feels like Microsoft and not just Intel but the whole processor industry are gearing up for another round of forced upgrades, this time regarding AI. The introduction of Copilot and the idea of an "AI" PC with Copilot key, and minimum AI hardware requirements, smacks of an industry desperate to sell fresh silicon, packing TOPs-generating neural processing units.

I doubt generic Windows is going to demand NPUs or a suitable GPU, but perhaps if you want to run Windows AI Copilot it will. It's potentially another step that makes a whole generation of non-AI PCs redundant and teetering on the edge of the landfill. It's bad enough that 32-bit systems struggle for support, but now the oldest 64-bit systems are also starting to run out of support if they lack suitable SSE instructions.

So, armed with a copy of Linux Mint Debian Edition and a helpful Jonni, we're taking a look at how you can breathe new life into old systems. A hardware update here, a software install there, and a PC that Windows left for dead can have a whole new life thanks to our Tux repair shop! If you have your own success stories, do let us know, and enjoy your new old PCs!

Neil Mohr Editor
neil.mohr@futurenet.com

Contents

REVIEWS

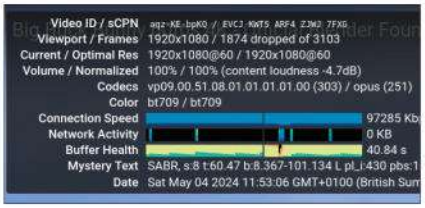
CREDIT: Magictorch

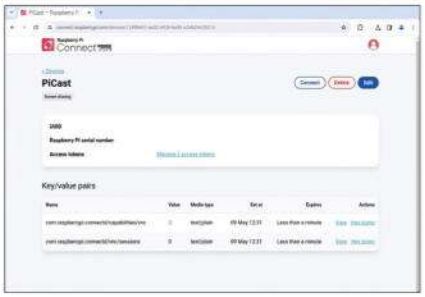
ROUNDUP

IN DEPTH

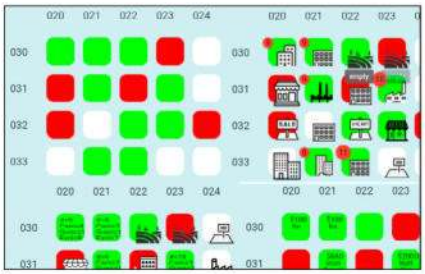
CREDIT: Zokara/Getty Images, Magictorch

Pi USER





CODING ACADEMY



REGULARS AT A GLANCE

TUTORIALS

ADMINISTERIA

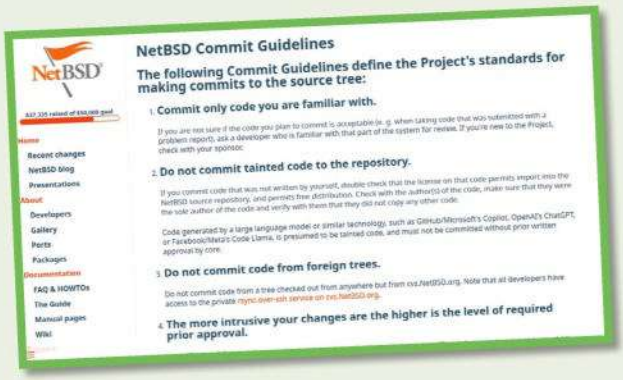
Newsdesk

THIS ISSUE: AI code drummed out of distros » Snapdragon supports Linux » KeePassXC commotion » No more Neofetch

DISTRIBUTIONS

Distros start to ban AI code

Generative AI continues to court controversy as Gentoo and NetBSD heavily restrict its implementation.



NetBSD's Commit Guidelines consider all **AI**-generated code to be tainted – unless the core developers provide an exemption for specific commits.

Despite the clear advantages of having AI create carefully crafted code, its use in major projects has been problematic. For instance, in March this year Nvidia found itself on the wrong end of a lawsuit. This centres on copyrighted material allegedly being used to train LLMs in the Megatron library for the tech giant's NeMo generative AI framework. In December 2023, the *New York Times* launched its own lawsuit against Microsoft and OpenAI, claiming that the newspaper's articles had been used to train ChatGPT and other models.

These legal wrangles likely won't go away any time soon, so it's unsurprising that back in February Gentoo Council member Michał Górny proposed a ban on using AI to develop the OS.

His justification for this was threefold: first he cited the above-mentioned copyright concerns. He also touched on quality concerns, given that in Górny's own words, "LLMs are really great at generating plausible-looking bulls**t." Finally, he pointed out that there are also ethical concerns, given its ownership by large corporations, which is "driving the ensh**tification of the internet".

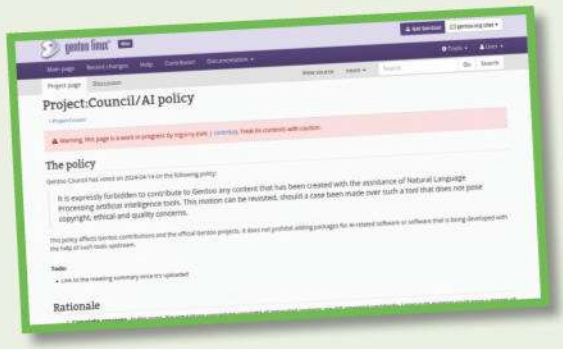
Górny's scatalogical post clearly struck a chord with the Gentoo Council – on 14th April, it voted unanimously to ban any contributions to the project made using natural language processing AI tools (see **https://wiki.gentoo.org/wiki/Project:Council/AI_policy**).

The resolution adds the important caveat: "This motion can be revisited, should a case been (sic) made over such a tool that does not pose copyright, ethical and quality concerns."

NetBSD also now has updated Commit Guidelines, which heavily restrict the use of AI-generated contributions. Provision two is entitled: "Do not commit tainted code to the repository." The guidelines go on to explain that code generated by an LLM or similar technology will automatically be considered to be tainted. As with Gentoo, though, the developers have given themselves an out, in that such code can be submitted with the prior written approval of the core development team.

Although no definitive policy changes have been made, in early May a discussion was also started in the Debian Project mailing list on this topic. Debian developer Tiago Vaz posted a message thanking everyone for sharing their thoughts but admits the community is "far from a consensus on an official Debian position regarding the use of generative AI as a whole in the project".

Regardless which way Debian or other major distros come down, these policies will be hard to enforce. It's often difficult to distinguish between human-written and AI-generated code.



Gentoo's policy on **AI** code is much more restrictive than **NetBSD**, but the council may revisit this decision in the future.

CREDIT: NetBSD, Gentoo Wiki

HARDWARE

Qualcomm on upstreaming to Linux

Patchsets for the Snapdragon X Elite SoC aim at Linux compatibility.

Qualcomm has a proven track record for supporting Linux on laptops for its 'premium' SoCs like the Snapdragon X Elite.

Qualcomm's Snapdragon X Elite has received favourable reviews for its performance in Windows running on Arm PCs. In mid-May, the company's developer blog also posted an announcement related to upstreaming Linux kernel support for its latest Snapdragon system on chip (SoC.)

The blog correctly states that this is nothing new. In the past, the company has cooperated with Lenovo, Arm and Linaro on the AArch64 laptops GitHub project, which provides Linux support for a number of its 'premium' SoCs.

The website also rightly states that Qualcomm is quick off the mark when it comes to Linux support; the initial patchset for Linux kernel support for the Snapdragon X Elite was announced on 24th October, just one day after the SoC itself was revealed to the world.

Qualcomm claims that the boot stack on Snapdragon X Elite supports standard UEFI-based boot. In tests, it was able to use GRUB to boot into Debian as well as dual boot Windows and Debian. In case there's any doubt, the company has even released an experimental, raw disk image for a Debian installer on GitHub. The link to this can be found in the blog post.

The same post also lists features that have been integrated into kernel 6.8/6.9 since the initial announcement. These include ADSP/CDSP support and multimedia clocks. Work is ongoing on merging features such as the USB host, GPU and camera for kernel 6.10/6.11.

Responses in the Redditsphere have been cautiously optimistic, though some users cite issues with Linux Wi-Fi compatibility, as well as a limited supported cycle for Qualcomm products.

Visit **https://bit.ly/snapdragonlinux** to read the full announcement.

SOFTWARE

APT 3.0 is coming

New APT solver comes with a number of changes.

For those who don't spend all their time poring over developer lists, the *APT (Advanced Package Tool)* solver is a component responsible for resolving package dependencies, conflicts and installation in Debian and variants like Ubuntu.

The solver ensures that whenever you install, upgrade or remove software, all dependencies are met and your system theoretically remains in a functional state.

In late May, Debian and Ubuntu Core developer Julian Klode made a blog announcement about the new *APT 3.0* solver.

He stated that *APT 2.9.3* now introduces the first iteration of the new solver, code-named solver3, which is fundamentally different from the old one: "Solver3 is a fully backtracking dependency-solving algorithm that defers choices to as late as possible."

All install requests now recursively mark dependencies with a single install solution. Any packages rejected due to conflicts cause their reverse dependencies to be transitively marked as rejected, provided they can't be solved by a different package. Any dependency with more than one choice is pushed to a priority queue.

The main noticeable difference is that solver3 retains manually installed packages. The *APT 3.0* solver also handles autoremove differently. Currently it only has information on the strongest dependency chain for each package, so it doesn't keep any packages that are only reachable via weaker chains. See **https://blog.jak-linux.org/2024/05/14/solver3/**.

CREDIT: Qualcomm

OPINION

A LIBRE SUMMER

Italo Vignoli
is one of the founders of LibreOffice and the Document Foundation.

"The Document Foundation has announced *LibreOffice 24.2.4 Community,* the fourth minor release of the free, volunteer-supported office suite for personal productivity in office environments, which is available at **www.libreoffice.org/download** for Windows, Mac OS and Linux.

The release includes many bug and regression fixes. Users who are not looking for the latest features and generally prefer a version that has undergone more testing and bug fixing should start planning the upgrade to the *LibreOffice 24.2* family.

LibreOffice is the only office suite with a feature set comparable to *Microsoft Office.* It also offers a range of interface options to suit all user habits, and optimises the space available on the desktop to put the maximum number of features just a click or two away.

Its biggest advantage is the LibreOffice Technology engine, a single platform for all environments: desktop, cloud and mobile. This allows *LibreOffice* to provide a better user experience and produce identical and interoperable documents based on both ISO standards for users concerned about compatibility, resilience and digital sovereignty, and the proprietary Microsoft formats."

OPINION

THE LONG HAUL

David Stokes
is a technology evangelist at Percona.

This month saw the release of *MySQL 8.4,* the first Long Term Support (LTS) release for *MySQL*.

MySQL 8.4 includes a range of bug fixes and changes. For example, the deprecated *MySQL* native password feature is no longer loaded by default, removing a potential security risk. The Clone plugin is more tolerant when working with different versions, and GTIDs have been extended, so groups of transactions can be processed, speeding up performance. We've seen the long-overdue removal of the terms Master and Slave, too.

The LTS approach delivers a version of *MySQL* built to last for at least two years, while Innovation releases will come out every quarter. The point releases for 8.0 onward often included changes that led to frustration, so LTS is a more reliable and steady alternative. However, those updates might take up to two years to join the LTS version.

MySQL admins often wait 12 months so bugs can be found and squashed. That would take us to May 2025, when we should be halfway through *MySQL 8.4*'s life cycle. This edition will undergo some heavy testing before being considered for production. Hopefully, Oracle's *MySQL* engineers have produced a rugged, durable, version that DBAs can commit to for the long haul.

SECURITY

KeePassXC kerfuffle

KeePassXC databases can be stored locally, offering better security, but it contains online features like browser integration and password sharing.

CREDIT: KeePassXC

Upcoming Debian Trixie implementation of KeePassXC won't offer optional features like browser integration.

Debian Developer Julian Klode is in the spotlight once again through making use of build options for the OS's implementation of *KeePassXC.*

Despite its dated-looking interface, this powerful password manager can create local encrypted databases to store user credentials.

Given the breaches of online password managers like *LastPass* and *Dashlane* recently, *KeePassXC* can offer a far more secure option.

The software itself does have online features like browser integration and password sharing, but these can be disabled. Klode chose to do so when uploading *KeePassXC* to Debian unstable for the upcoming Debian 13 (Trixie) release.

He justified his decision by pointing out the XZ backdoor, which was only uncovered by chance by a skilled developer. Given optional features are less likely to be reviewed, he explained that there's a greater chance they'll have vulnerabilities.

At the time of writing, Debian users can still install *KeePassXC-Full,* which unlike the minimal *KeePassXC*, has online functionality.

GRAPHICS

Framework optimises screen

Invisible pixels promised.

In late May, Framework announced the latest Laptop 13 series of laptops, which offers an optional 13.5-inch 2,880x1,920 120Hz display on both Intel and AMD systems.

The new 2.8k display option at 256ppi has led the company to claim that pixels will be effectively "invisible from a normal viewing distance". While we will leave such judgments to LXF's hardware reviewers (*who?–Ed*), there's no doubt that this new resolution will allow for a more streamlined experience in Linux through 2:1 display scaling. You can read more at **https://bit.ly/framework13linux**.

The latest Framework display has an optimised screen with 2:1 display scaling. This should avoid fractional scaling when running Linux.

SOFTWARE

Neofetch is dead

Creator of utility archives repo to take up farming.

On 26th April, Dylan Araps, creator of *Neofetch,* archived the GitHub repository for the project.

This didn't come as much of a surprise to the community, as the utility has only been updated infrequently in the past few years. Araps updated the Readme file on GitHub to say that he has "taken up farming".

Currently, the utility is available in the repositories of popular distros such as Ubuntu. If you need a futureproof method to display system information in a colourful way, we recommend *Hyfetch,* a well-maintained direct fork of *Neofetch*.

There are many viable forks of Neofetch, including Hyfetch (pictured), which can display your system information using pride colours.

CREDIT: Framework, Canonical

Distro watch

What's behind the free software sofa?

LINUX LITE 7.0

This beginner-friendly distro is based on the latest Ubuntu LTS release and uses the Xfce desktop environment. According to its developers, the final release of version 7.0 (code name Galena) follows the theme of maturity, to reflect the fact that Linux Lite has been around for 12 years. This means there have been minor revisions to the welcome screen, which also incorporates new install slides. *Thunar* file manager also now supports split view. Learn more at **www.linuxliteos.com**.

Linux Lite has a tweaked file manager and welcome screen.

EUROLINUX 8.10

EuroLinux is designed for enterprise environments and incorporates large amounts of RHEL code, making it compatible with its parent OS and other distros like Oracle Linux. Version 8.10 was released on 27th May, preceded by version 9.4 on 13th May. The eccentric numbering system aside, version 8.10 (code name Bucharest) includes a number of new modules, such as *MariaDB, Nginx* and *PostgreSQL*. Critical packages such as GCC have also been upgraded. You can discover more at **https://euro-linux.com**.

EuroLinux is compatible with **RH**EL and derivatives.

ULTRAMARINE LINUX 40

Ultramarine is based on Fedora but incorporates extra repos such as RPM Fusion. It also bundles various multimedia codecs. Its flagship edition uses the Budgie desktop environment, with Gnome and Plasma as alternatives. The latest version 40 (code name Umbrella) also offers an Xfce spin coupled with the Materia theme. Ultramarine images are also now available for Pi. The developers plan to support more devices like Chromebooks and Macs in future. See **https://ultramarine-linux.org**.

Ultramarine is now available with Xfce and supports the **P**i.

ARMBIAN 24.5.1

Armbian is designed for ARM development boards, such as the Banana Pi. The Ubuntu version of the latest release (code name Havier) is based on Noble Numbat, complete with the Gnome desktop but minus Canonical-specific code like snapd. It also bundles the latest versions of *VS Code, Thunderbird, Firefox* and *Chromium*. Rolling builds of Armbian are also available based on Debian Testing/Trixie or Ubuntu Oracular, enabling users to test bleeding-edge features. Find out more at **www.armbian.com**.

Various Armbian builds are based on **U**buntu and **D**ebian.

GHOSTBSD 24.04.1

GhostBSD is touted as a more user-friendly version of FreeBSD, being based on the latest stable release. The current OS has an extensive changelog, including an upgraded MATE desktop (v1.28.1). The installer has been revised to stop the root password being changed if users update their admin credentials. GhostBSD now incorporates the desktop version of *Signal Messenger*. It also loads the ig4 module by default for better touchpad compatibility. See **www.ghostbsd.org**.

Ghost**BSD**'s tweaks include better touchpad support.

OPINION

FEEL THE G-FORCE

Daniel Morin
is a senior software developer at Collabora.

"The latest major release of GStreamer, version 1.24, was made available a few months ago, the culmination of 13 months of meticulous and dedicated effort by the community. No fewer than 231 developers contributed to this release.

GStreamer remains the pre-eminent open source multimedia framework, and with this release it rises to new heights in embedded and network streaming. We also took a bold step forward by introducing a machine-learning analytics foundation, enabling GStreamer to be the go-to choice for building robust analytics pipelines.

A stride forward has also been made with the creation of the GStreamer W3C Media Source Extensions (MSE) library. This introduces a GStreamer API aligned with the W3C Media Source Extensions specification, empowering applications to seamlessly integrate existing code for media processing with GStreamer without the necessity of a web browser engine. Built upon WebKit's robust implementation of the Media Source spec, it offers a versatile foundation, with a C API similar to the JavaScript API available in browser code.

Our next steps will be to add AV1 support to visl and expand analytics support with multi-modal analysis, TFLite inference, and AnalyticsMeta2Onfiv."

OPINION

ALPHA & OMEGA

Jon Masters is a kernel hacker who's been involved with Linux for over 22 years, and works on energy-efficient Arm servers.

"This month's kernel hardware victim was support for early Alpha CPUs (pre EV56), which featured in some of the mid-'90s DEC Alpha Workstations. Alpha was introduced in the early '90s as a modern RISC-like alternative to x86. It featured a large number of registers and an (infamously) relaxed memory model that brought higher performance but made programming for it more complicated.

Alpha would later be extremely influential upon contemporary architectures, such as Armv8. The ghost of Alpha lives on, both in those of us who still own the hardware, and in the influence the need to support it still has upon Linux's memory model.

Linus himself remains fond of Alpha, noting that he "dearly loved" it "back in the days", but "if you want to run museum hardware, maybe you should use museum kernels". In this case, he's removing older Alpha support because these are the last chips that didn't have byte-level access by default, which was complicating some kernel code for little benefit. Still, my own "museum" (aka basement) will continue to house my Alpha hardware for the foreseeable future."

Kernel Watch

Jon Masters summarises the latest happenings in the Linux kernel, so that you don't have to.

Linus Torvalds has announced the release of Linux 6.9, noting that "the whole release has felt pretty normal". The new kernel adds support for pidfds by individual threads (not just processes) – simplifying race-free process management in userspace – the new FRED (Fast Return and Event Delivery) mechanism present on future Intel processors (intended to simplify and speed up interrupt delivery), host support for AMD's Confidential Compute feature SEV-SNP (Secure Nested Paging), along with a large number of other features. As usual, KernelNewbies has an excellent summary of the new features, which you can find here: **https://kernelnewbies.org/Linux_6.9**.

With the release of Linux 6.9 came the opening of the merge window (period of time during which disruptive changes are allowed to be made to the kernel code) for what will be Linux 6.10 in a couple of months. The merge window was open for the customary two weeks, and was then closed with the release of Linux 6.10-rc1. In addition to removing support for older Alpha chips, 6.10 adds a lot more Rust language infrastructure (including initial support for Rust on RISC-V), initial – but not yet enabled – support for the ntsync Windows NT (aka modern Windows) sychronisation primitives (similar to futexes but different enough to require a special subsystem, which should speed up games running under Windows emulation in *Wine* by as much as 150%), and the new mseal system call that effectively locks the address space of an application against future changes.

HANGING AROUND

"**A** few regressions in 6.9 were highlighted, including an **A**B-B**A** deadlock in some code used to manage **LED**s on network cards that caused unexpected hangs."

A few regressions in 6.9 were highlighted by Thorsten Leemhuis, including an AB-BA (competing spinlocks acquired by two processes in the opposite order) deadlock in some code used to manage LEDs on network cards that was apparently causing a bunch of users to experience unexpected hangs when booting 6.9. LXF

» ONGOING DEVELOPMENT

A large amount of work is ongoing to refactor the Linux memory management subsystem, including through further adoption of folios, the replacement for traditional page structures that track the fundamental minimal sized chunks through which hardware manages memory. The end goal is to make Linux memory management more flexible, and in particular scalable for the very large machines it is used on today. Linux Weekly News (LWN) has a good summary of the sessions on folios that were just held at this year's Linux Storage, Filesystem, MM & BF Summit. You can find that at **http://lwn.net**.

The KernelCI community has announced a "switch to new testing architecture", along with a timeline during which it will migrate over to a newer system. This only impacts those developing new kernel features who are interested in monitoring the results of these various automated kernel tests. More info is at **https://kernelci.org/blog/posts/2024/strategic-updates/**.

Planning continues for the upcoming Linux Kernel Plumbers Conference, along with the assorted other microconferences and events that will be co-located. This year's conference will be held in Vienna in mid-September.

Answers

Got a burning question about open source or the kernel? Whatever your level, email it to **answers@linuxformat.com**

Neil Bothwick swallowed a stick of RAM to improve his memory.

Q Timeshift takeover

I am apparently out of space on my SSD, which is odd and shouldn't be the case. I tried to clear up some space (around 15GB) and nothing seems to free up room. I then noticed that *Timeshift* is running, because the logo is visible on the bottom-right side of my screen (it usually isn't there). When I hover over the logo it says, "A *Timeshift* system snapshot is being created." It doesn't let me open *Timeshift* or any program really. When trying to open *Timeshift*, it says "Scheduled snapshot in progress…"

I went into my recycling bin to try clearing it for space and it doesn't let me do that either. It says, "Failed to delete this item from trash, do you want to skip it?" After learning that clearing space or my recycling bin won't do anything, it's been over a day, and I haven't turned my PC off because I'm scared it won't turn back on.

Edward Walters

A It appears that *Timeshift* is stalled mid-backup because the filesystem is full. You could kill *Timeshift* but that would most likely leave you with a corrupted backup. A safer approach would be to free up some space to allow *Timeshift* to complete the current backup. Start by deleting downloaded package files from software installations. To do this, fire up *Synaptic,* go to Preferences > Files and press Delete Cached Package Files. You may also want to select Delete Downloaded Packages After Installation to stop the cache filling up again.

Now open the file manager, set it to show hidden files and look at the **.cache** directory in your **home**. This contains all sorts of files that are not necessary but can speed things up a little. Begin by deleting the **thumbnails** directory, and then delete the rest if you are still short on space.

You should also get rid of any large data files that you have hanging around that you know are not necessary for the system. Things such as videos and photos can be deleted or moved to an external drive to gain space. If all of this is enough to allow *Timeshift* to complete the currently running backup, you are free to enter *Timeshift* and remove older backups to regain space.

If not, you will have to force *Timeshift* to close with this terminal command:

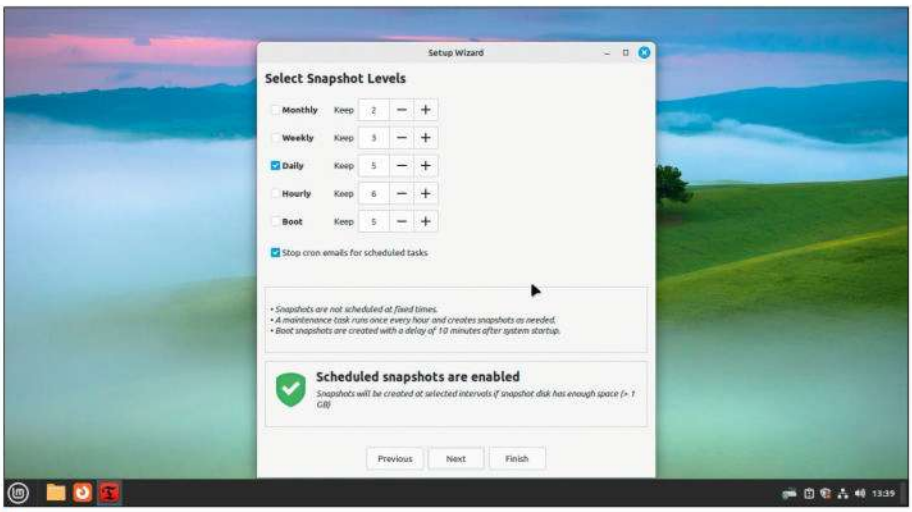
```
$ killall timeshift
```

This sends a TERM signal to the *Timeshift* process, asking it to shut down cleanly. If you get an error saying you don't have permission to do this, prefix the command with `sudo`. Give *Timeshift* time to shut down and clean up after itself, then try opening it. If it will not close cleanly, you need to be a bit more brutal with:

```
$ sudo killall -KILL timeshift
```

This sends the KILL signal to the *Timeshift* process, terminating it immediately – and uncleanly. Either way, you should then open *Timeshift* and delete the most recent backup, because it will be incomplete at best, corrupt at worst, as well as some of the older ones to free up space. Now create a new backup.

In future, it is best to have *Timeshift* back up to a different drive, say an external one. While backups on the same drive are useful when you have an oops moment and need to backtrack, they are no help if you have a drive failure.



Saving **Timeshift** backups **to the same disk** can **soon** fill **it** up, **especially if you keep a lot of them**.

Q Whose disk is it?

I have an internal NVMe disk and when I mount it using a file manager, it mounts to **/media/<disk-id>**, which is fine. The problem is that after mounting, the directory is owned by root:root and the permissions are 0700, which means that the user who just mounted the filesystem can't actually access it. I suspect there is a udev rule somewhere that controls the mount options or falls back to some default values.

Leah Brookes

A The solution to this depends on the filesystem(s) used on the NVMe disk. With Linux filesystems, the ownership and permissions are stored in the metadata of the filesystem itself. If this is used purely for data for your user, as opposed to a filesystem containing the operating system, you can set the drive to be owned and writable by your user with the following commands:

```
$ chown username: /media/disk-id
$ chmod 755 /media/disk-id
```

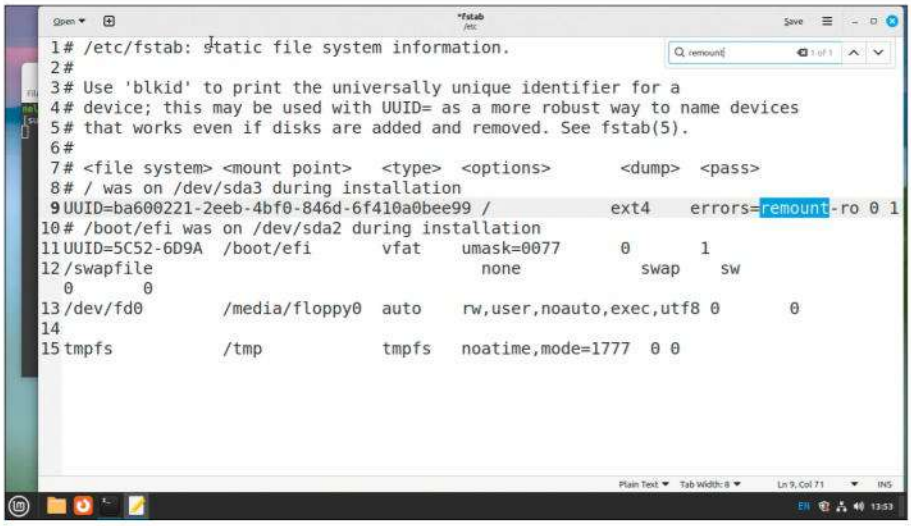
You may also need to alter ownerships on the files on the drive. You can set them all to be your files by adding the **-R** option to the *chown* command, but only do this with data files.

If the drive had a Windows filesystem on it, this approach would not work as Windows filesystems have no way of

remembering Linux permissions. Because this is a fixed drive, presumably mounted at boot time, the easiest way is to do this from **/etc/fstab**. For a FAT filesystem, add this line to your **/etc/fstab**:

```
UUID=disk-id /mnt/nvme1 vfat uid=your user,gid=yourgroup,umask=022 0 0
```

Change `vfat` to `ntfs` if you have an NTFS Windows filesystem. Because the Windows filesystems have no concept of Linux users and groups, the **uid** and **gid** options set all files to be owned by the given user and group. The **umask** option sets permissions to 644 for files and 755 for directories, which is usually what you want. The mount point given in the second field can be wherever you want but the directory must already exist – **/media** is not recommended because that is mainly for transient mount points for removable devices. This mounts the filesystem automatically when you boot; if you don't want that, add **noauto** to the list of options in the fourth field. And in that case, you may also want to add **user** to the options, otherwise only the root user is allowed to mount the drive.



This remount on error setting in fstab can cause your disk to become read-only if an error is detected. Fixing the error is preferable to disabling this setting!

Q Locked out of home

I am using Linux Mint 21.2 Mate, which is unmodified since install except for updates. The contents of the home folders have become read-only and I cannot copy them. Upon reboot, I get full user access again, but *Caja*'s copying fails and reports read-only again, and the icons are padlocked.

Toby Curtis

A A filesystem going read-only is usually in response to an error. When the filesystem detects anomalous or corrupt information, it sets itself to be read-only to prevent any further damage. If you are using the default arrangement of **/home** being on the root filesystem, you need to boot a live distro to fix this, because repairs usually require the filesystem to be unmounted. Run this terminal command to see which device contains your home directories:

```
$ df -h /home
```

This returns something like **/dev/sda3**. The first step is to search the system journal for any errors relating to that device. To show all error messages logged by the system since the last reboot, in your terminal run:

```
$ journalctl -b -p err
```

This shows the information in your default pager, usually *less*, where you can use the Spacebar to scroll down as required, or search by pressing `/` followed by a search string, say `/sda3`.

You can also pipe the full log since boot through *grep* to find all mentions of your filesystem:

```
$ journalctl -b | grep sda3 | less
```

There could be more than a screenful, so we pipe it through *less* once again. If nothing obvious pops up, or if you see a clear indication of filesystem corruption, the next step is to boot from a live distribution – the Mint installation disc or USB is fine. Now open a terminal and check the filesystem with *fsck:*

```
$ sudo fsck -f /dev/sda3
```

Use the correct device for your system, of course. The **-f** option forces a check even if the filesystem is not currently marked as needing one, which may be the case immediately after a reboot. Hopefully, *fsck* will find and be able to fix any problems. If not, you may have a physical problem with the hard disk, which is something that neither *fsck* nor a reinstall can fix.

For a physical health check on your disk, you need to install the *smartctl* program in the normal way, and to have SMART enabled in the BIOS of your computer. To test a drive, run:

```
$ smartctl --test=long /dev/sda
```

This runs the long test in the background and tells you how long it should take. After which you can see the results with:

```
$ smartctl --log=selftest /dev/sda
```

You can also get a quick health summary for a drive, after at least one test has been run, with:

```
$ smartctl --health /dev/sda
```

» A QUICK REFERENCE TO... TMPFS

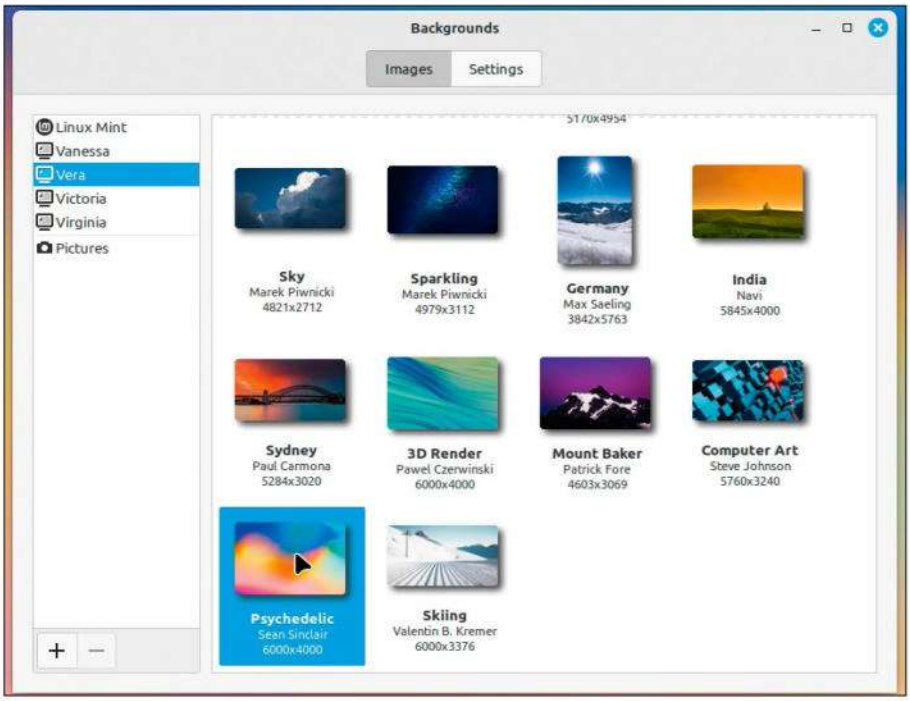
Linux has more filesystems than you can shake a stick at. There are the traditional disk-based filesystems, such as ext2/3/4, XFS and so on. Then we have the all-in-one filesystems and volume managers, such as ZFS and Btrfs, not to mention the virtual filesystems used in **/proc**, **/sys** and **/dev**. So, here's one more: tmpfs. This is a filesystem that exists only in memory. When the computer is switched off, its contents are gone – for ever. This makes it useful for data that you don't want hanging around. It is most commonly used to mount the **/tmp** directory, because that is supposed to be wiped when you reboot anyway. Because the files are stored in memory only, tmpfs is fast, much faster than a spinning disk and also faster than an SSD. You create a tmpfs from the command line with:

```
$ sudo mount tmpfs /mountpoint -t tmpfs
```

Or you can create one from **/etc/fstab** with a line like:

```
tmpfs /tmp tmpfs size=50% 0 0
```

The **size** option sets the maximum size of the filesystem, either in bytes or, as here, as a percentage of total RAM – 50% is the default. This doesn't mean that mounting a tmpfs will gobble up half of your memory – this is the maximum size; it only uses as much space as it needs to store its contents. When you unmount the filesystem, all memory is returned.



Continually-changing wallpapers do not sit well with Gnome-like desktops, but you can script frequent updates.

If this reports errors, your drive is on the way out and should be backed up and replaced as soon as possible.

Q Missing link

Why do symlinks do not contain the paths? Everywhere you read about symlinks, you are told the symlink file data contains the target file path and name, but when you check the file, it contains only the name. Where is the path stored? I am not interested in the actual command to find the path. I am interested to know where the path is actually stored and maybe an explanation why is not together with the name.

Jonathan Long

A You can see the target of a symlink with `ls -l` or *readlink*. This may be what you are using and it can show either the full path to the target or just the name, depending on how the symlink was set up. This is because a symbolic link can contain either an absolute or a relative reference to its target.

To see this in action, create a temporary directory within your **home** directory and a file within it:

```
$ mkdir -p ~/tmp
$ cd ~/tmp
$ date >f1
```

Now run these commands to create three symbolic links to that newly created file and list them:

```
$ ln -s f1 l1
$ ln -s ~/tmp/f1 l2
$ ln -s ../tmp/f1 l3
$ ls -l l?
```

All three point to the same file, but **l1** contains just the name, so it only points to a file in the current directory, whereas **l2** has an absolute path – **l3** is a relative path. This means if you move these links without moving the target, **l2** will still work while the other will break.

You may also see reference to fast and slow symlinks. These relate only to how the target path is stored. If the path is short enough, the target is stored in the file's metadata – that is a fast link. That would explain your not being able to see where the information is stored. If the path is too long, it is stored in a block on the filesystem. In this case, the symlink is a plain text file containing the path, but the system treats it as a link because the filesystem marks it as such. The use of fast and slow symlinks depends on the filesystem in use.

Q Another GIF in the wall

I would like to be able to update my wallpaper continuously with a Python script and OpenCV, and I would rather not do this by simply writing to a file constantly. I have a script that does this just fine. What I would like it to do is send the image data directly to the part of the operating system that renders the wallpaper without saving the image file, but I have had no luck finding information about how to do that .

Owen Fowler

A The wallpaper system used by desktops is not really designed for that. It works around the method of loading a static image from disk and holding that in memory until it is changed. There are plugins that can use a video file as animated wallpaper, with varying degrees of reliability. The KDE Plasma one seems the best, but you are running Cinnamon. If you can find an animated wallpaper plugin that works for you, you could create a pipe with *mkfifo,* send the video information that and set the pipe file as the source for the wallpaper.

It rather depends on whether you actually need a continuous update or whether updating once a minute, say, would suffice. In the latter case, you can have your script save the image to a file in RAM using a tmpfs filesystem. Some distros put **/tmp** on a tmpfs by default; with Mint, you need to add this line to **/etc/fstab** and reboot:

```
tmpfs /tmp tmpfs noatime,mode=1777 0 0
```

Alternatively, there is already a tmpfs at **/dev/shm** that you could use. Once you have done this, you can have your script save to a file in the tmpfs, which will be stored in RAM, and then tell Cinnamon to reload it with this command:

```
$ gsettings set org.cinnamon.desktop.
background picture-uri file:///tmp/
background.png
```

Saving multiple files in a tmpfs will gradually use up your RAM, so either use the same name each time or delete the older files when saving new ones. There is one potential side effect of using a file in RAM: it may not be there when the desktop loads, so you may just get a blank background until the first run of your script. LXF

GET HELP NOW!

We'd love to try to answer any questions you send to answers@linuxformat.com, no matter what the level. We've all been stuck before, so don't be shy. However, we're only human (although many suspect Neil is an poorly run AI), so it's important that you include as much information as you can. If something works on one distro but not another, tell us. If you get an error message, please tell us the exact message and precisely what you did to invoke it.

If you have, or suspect, a hardware problem, let us know about the hardware. Consider installing *hardinfo* or *lshw*. These programs list the hardware on your machine, so send us their output. If you're unwilling, or unable, to install these, run the following commands in a root terminal and send us the system.txt file, too:

```
uname -a > system.txt
lspci >> system.txt
lspci -vv >> system.txt
```

Mailserver

WRITE TO US

Do you have a burning Linux-related issue that you want to discuss? Write to us at *Linux Format*, Future Publishing, Quay House, The Ambury, Bath, BA1 1UA or email letters@linuxformat.com.

Quantum Leap

Seeing as Linus Torvalds wrote programs and games (including *Pac-Man* clone *Cool Man)* for the Sinclair QL, perhaps *Linux Format* should do an article about the QL, celebrating 40 years since its release, and maybe include an interview with Linus about how the QL influenced his computing journey. This website is an excellent source of information: https://dilwyn.qlforum.co.uk.

Alan

Neil says...

It does look as though the Sinclair QL slipped through the net when we ran our classic emulation series. I believe Les found that it was supported by the ZX Spectrum emulator he covered, and because he had no personal experience with the QL, he decided to skip over it. We're a shambles!

Linux, BSD and Mac OS are POSIX compatible, so you'll find that many tools work across all three.

Free stuff

You spend a lot of time talking about Linux – I guess the clue is in the magazine's name – however, there are a lot of other open source operating systems out there. Does anyone actually use any of them as a daily driver? Does anybody use ReactOS or that kind of stuff?

Bob Jeffries

Neil says...

So, I'm not going to apologise for just covering Linux operating systems – there's enough content there to last us until the heat-death of the universe! However, you've asked a good question – and who really knows? Of course, the various BSDs are in use commercially (widely) and (to a much lesser extent) on personal systems – don't try to argue that Mac OS is a BSD. Moving outside of BSD, though, I think you're exploring niche communities, which isn't a bad thing – they're very interesting and that's why people get involved. So, for example, ReactOS is attempting to reverse-engineer a Windows-compatible OS, Haiku is keeping BeOS alive, and Open Indiana is doing the same with Solaris and so on. The last *Roundup* that looked at alternative free operating systems was in **LXF251**.

Dav1d is the open source project for efficient AV1 decoding.

High-confusion encoding

I don't see you writing much about video, and now that my phone is recording in 4K, I'm wondering what the best way of encoding its footage would be. Back in the day, I'd be using DivX and MP4, but when it comes to this HEVC, I'm a bit clueless. I'm guessing *Handbrake* is still the best tool to use.

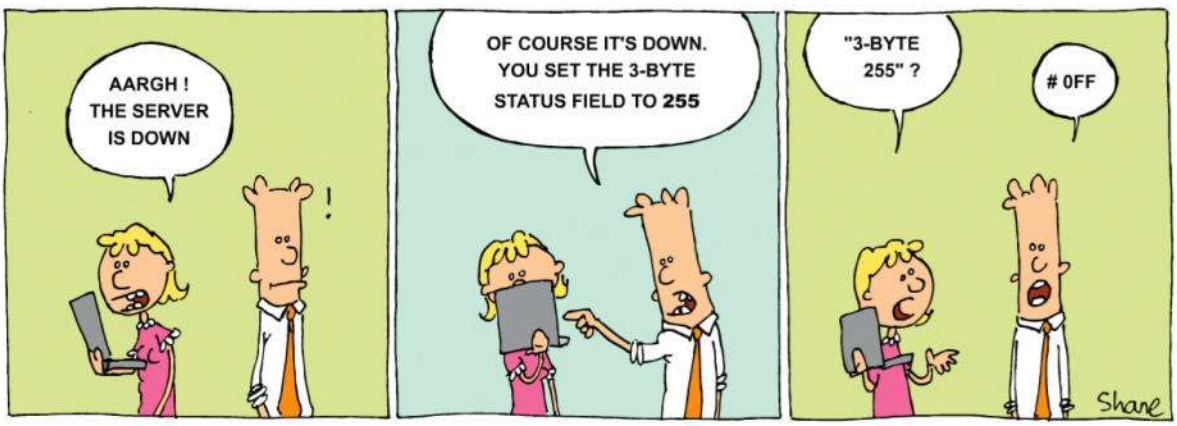
Tim Madson

Neil says...

The situation is probably worse than you think in terms of different encoding options out there. The

The Sinclair QL was a failure, targeting business users who were already wooed by the IBM PC.

Helpdex



old DivX/h.264 and now h.265 were always locked up in some sort of software patent, which is why there are x264/265 open source implementations. Because of that Google, Netflix and others struck out to create from-scratch patent-free (though there are counter claims) open codecs such as VP9 and AV1. See the dav1d project (**www.videolan.org/projects/dav1d.html**) as an example from the VideoLAN people. But generally, yes, *Handbrake* is up to date, although *ffmpeg* is where the command-line action is.

SAMBA

When it works, which it usually does, Samba is great for network shares.

Samba mania

I have been trying for several months to get *Samba* working on my Fedora Server installation. I had it working fine before I did a reinstallation, when I was trying to figure out why it was not connecting or something (I don't fully remember what the issue was now). I have tried to follow multiple tutorials and only get as far as it showing an empty share. I don't know whether I messed up when trying to split up **smb.conf** and **shares.conf**. I am pretty clueless when it comes to networking, but I just need to be able to connect to the server's network share with Windows and Ubuntu.

Sara Johnson

Neil says...

Months? That's a bit crazy for something that should work out of the box. We ran a tutorial in **LXF316** on the basics of using *Samba,* but I doubt that's going to help in your situation. We're not sure that splitting the conf was the right thing to do. Take a look at the official Fedora Samba documentation: **https://docs.fedoraproject.org/en-US/quick-docs/samba/**. It'll take you through adding a share to the **smb.conf** file and go on to group shares.

It's important to restart the *Samba* service after any changes are made with `sudo systemctl restart smb`. We've had sharing issues in the past because one service was trying to use SMB v1, which is now verboten by default, as is v2 I think too. LXF

» LETTER OF THE MONTH

6809 forever

The 8-bit 6809 MCU from Motorola had the most programmer-friendly architecture I've ever seen on any chip, 8-bit or otherwise. It was a joy to program in assembly for it. The primary programming language was Basic09; it was far beyond most Basics and included several structured programming concepts, such as **WHILE**, **UNTIL**, and **FOR** loops and nested **IF – ELSE** to mention a few.

The interpreter would fit into 8KB. The primary OS, OS9, was re-entrant, position-independent, interrupt-driven and multitasking. You could fit the OS into 8KB of ROM. It borrowed concepts from Unix, such as tree-structured directories, everything is a file, and the ability to pipe the output of one program to the input of another.

OS9 Level II added memory management so that the 8-bit chip had a 24-bit address space. The Radio Shack CoCo used that chip. I ported an APL interpreter to OS9. Alas, like many Motorola products to come, it was 'a day late and a dollar short'. The Z80 and CP/M already owned the 8-bit space, and the world was already moving to 16-bit and the IBM PC. If Motorola had skipped the 6809 and brought the 68000 out a year sooner, IBM could well have chosen the 68000 for its PC (it was its first choice) and the PC OS might well have been OS9 for the 68k aka OS9-68000. How different the computing landscape might have been!

Greg Morse Richmond, Canada

Neil says...

It's fantastic to hear such interesting memories from our readers. I hadn't come across the Motorola 6809 before, so it's good to hear about it, though I had a good go at assembly on the 68000 and Amiga, plus I did a writeup of the PowerPC architecture at university. It's amazing how streamlined those architectures were versus x86.

The Motorola 6809 was an advanced 8-bit processor from 1978.

CREDIT: CPU Collection Konstantin Lanzet

shane_collinge@yahoo.com

TAKE YOUR LINUX KNOWLEDGE FURTHER

Enhance your knowledge with in-depth projects and guides

FUTURE

Ordering is easy. Go online at:

magazinesdirect.com

Or get it from selected supermarkets & newsagents

Ryzen 7 5700X3D

Paul Alcorn is feeling more powerful than ever with AMD inside!

SPECS

Socket: AM4
Process: 7nm TSMC
Cores: 8
Threads: 16
Clock: 3GHz (4.1GHz boost)
Cache: 512KB L1, 4MB L2, 96MB L3
Unlocked: No
GPU: N/A
Mem: DDR4-3200, ECC support, 2-channel
PCIe: v4 20 lanes
TDP: 105W (142W PPT)

AMD's X3D line of processors has taken the gaming world by storm, delivering leading gaming performance at every price point they compete in, and the £220 Ryzen 7 5700X3D steps in to lower the pricing bar for entry. That's a win for enthusiasts upgrading existing AM4 systems and even for those looking to build a budget gaming rig. The Ryzen 7 5700X3D offers the highest gaming performance in its price bracket, even unseating Intel's more expensive £310 Core i5-14600K.

As with AMD's other X3D chips for AM4 motherboards, the Ryzen 7 5700X3D is based on the previous-gen Zen 3 architecture, so it isn't quite as performant in productivity applications and won't rank as high in our CPU benchmark hierarchy – you'll be better off with the Ryzen 5 7600X if you're interested in a better all-rounder. However, the Ryzen 7 5700X3D drops into affordable AM4 motherboards.

The Ryzen 7 5700X3D is basically a down-clocked Ryzen 7 5800X3D, losing 400MHz off its base and boost frequencies. All other details remain unchanged, including the voluminous 96MB of game-boosting L3 cache. And, in effect, it offers two more cores than the 5600X3D for the same price.

Life in the fast lane

The Ryzen landscape is divided into two lanes: you can choose between newer Zen 4 chips that drop into AM5 motherboards and use DDR5 memory, or a Zen 3 chip that drops into older AM4 motherboards and uses DDR4 memory.

AMD has continued developing its Zen 3 chips, and says it will continue to support AM4 for the foreseeable future as a value-centric platform for the lower end of the market. And that has certainly come to fruition – AM4, which debuted back in 2017, is in its eighth year of service, the longest-supported modern socket that still receives new processor additions.

It comes without a cooler, so you have to factor that into your build pricing. However, the chip is fairly easy to cool with value-centric coolers – it beats the 181W Core i5-14600K in gaming, but at a much lower peak power draw. The Ryzen 7 5700X3D also doesn't have integrated graphics – you also need a discrete graphics card for gaming.

The Ryzen 7 5700X3D is optimised for gaming, but the 3D V-Cache tech results in lower clock speeds that hamstring performance in productivity apps. The chip also leverages the previous-gen Zen 3 architecture, so app performance is particularly lacklustre compared to newer Ryzen 7000 and Intel chips.

The Ryzen 7 5700X3D is a whopping 30% faster than the Core i5-14400 in gaming, but it has a very clear disadvantage in productivity applications. The Core i5-14400 is 10% faster in multithreaded work and 27% faster in single-threaded applications than the 5700X3D, so it offers a much more balanced profile overall.

Ideal for gamers, but not so much anyone else...

If you're looking for a more balanced Ryzen chip that's even better than the Core i5-14400, the £209 Ryzen 5 7600X fits the bill. This chip is 6% faster than the Core i5-14400 in single-threaded work and 3% faster in multithreaded applications, so it's a solid all-rounder if you prize performance in productivity workloads. It's also 14% faster in multithreaded and 37% faster in single-threaded workloads than the Ryzen 7 5700X3D.

The Ryzen 7 5700X3D consumes a peak of 117W, an impressive 100W less than the Core i5-14600K that it actually beats in gaming workloads (but trails in productivity applications). Notably, this is much lower than the theoretical PPT (peak) of 141W for AMD's 105W TDP rating. As expected, the Ryzen 7 5800X3D with higher clock rates pulls a few watts more in most workloads, highlighting that there isn't much difference between the two chips in day-to-day use.

The Intel Core i5-14400 can't compete with the reduced pricing of the modern Ryzen 5 7000-series models, but the Ryzen 7 5700X3D pressures Intel's line-up from the other end of the spectrum. LXF

VERDICT

DEVELOPER: AMD
WEB: www.amd.com
PRICE: £220

FEATURES	8/10	EASE OF USE	9/10
PERFORMANCE	8/10	VALUE	9/10

If you're interested in gaming, the **Ryzen 7 5700X3D** is the go-to chip for budget builds, particularly for AM4 upgraders.

» Rating 9/10

LibreELEC 12.0

Nate Drake is impressed with the freedom that the latest version of LibreELEC (code name Omega) offers to cosy up to Kodi.

IN BRIEF

Although the setup was a little clunky, post-install LibreELEC runs *Kodi* flawlessly. It remains the perfect OS to play and stream your favourite content from one place.

SPECS

CPU: 1GHz
Mem: 1GB
HDD: 500MB
Builds: AMD64; ARMhf, ARMv7, ARM64

LibreELEC has been in active development since 2016, when a number of OpenELEC programmers decided to fork the project.

According to the main website, its purpose is to provide "just enough operating system" to run the *Kodi* media player, which is reflected in the fact that it runs happily on single-board computers such as the Raspberry Pi 4 and 5.

This said, the project wiki advises against trying to run LibreELEC on legacy hardware, as "just enough" doesn't always stretch to older device drivers.

There's no bootable ISO but interested parties can download a disk image to write to a bootable USB.

Although LibreELEC isn't really designed to be run via a hypervisor, there's a virtual machine (OVA) image available for testing purposes, which we used.

Given how lightweight the OS is, perhaps it's not surprising that the wiki doesn't seem to list minimum system requirements. The specs listed *(left)* are the result of our tinkering in a VM, but we had to increase the available video memory from 16MB to 64MB.

Upon boot, we were offered the choice to install LibreELEC, boot it in live mode, or run as is. After choosing Run, the OS needed to repartition the disks before restarting, prompting us to choose Run again.

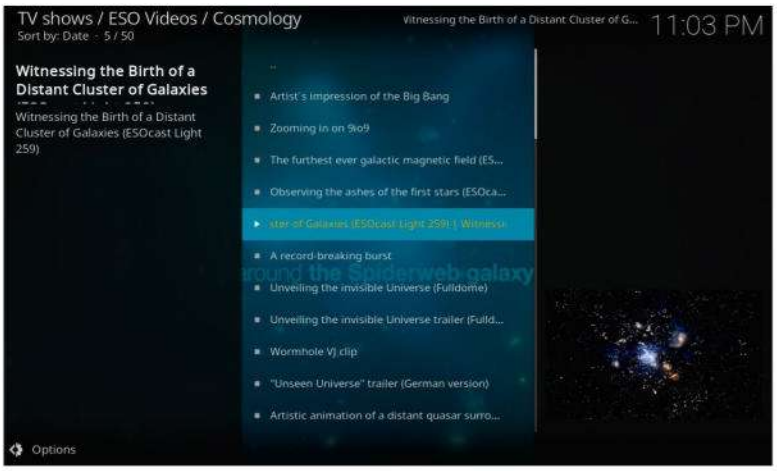
Once we overcame these teething troubles, we were greeted by the friendly *Kodi* setup screen. This simple interface enables you to set up the OS in moments by choosing the system language, a more imaginative hostname (the default is 'LibreELEC'), as well as configure network settings and remote access via SSH and/or *Samba.*

On first load, the system helpfully informs users that their movie and TV show libraries are empty but these can be populated by going to the Files section.

However, LibreELEC offers far more than a simple media player. You have access to all the official *Kodi* add-ons, as well as the LibreELEC add-ons repository. These include *Kodi* PVR (personalised video recorder) clients, servers, screensavers and visualisations.

Our initial attempts to install the *Demo* PVR and even an unofficial one for Pluto TV were successful, but after this LibreELEC seemed unable to connect to the internet. We were able to resolve this issue by removing and reading the virtual machine image.

In fairness, the comprehensive LibreELEC forums explained that the image in question is designed to run in *VMWare,* rather than *VirtualBox,* our program of choice. For this reason, we have not deducted any points from our Performance ratings in this review.



LibreELEC has its own add-ons repository. You can use this to extend the OS's functionality – to access streaming services, for example.

On the plus side, add-ons are neatly arranged into categories such as Games, Videos and Music. Our attempt to load videos from the ESO (European Southern Observatory) were much more successful, with the relevant add-on downloading in seconds.

LibreELEC in general is lightweight – our virtual hard disk took up less than 250MB. This is reflected in the ease with which you navigate between menu options. We were particularly impressed to see that each settings menu defaults to Standard view, but you can toggle this to Basic, Advanced or Expert to view fewer or more options depending on your experience level.

Given the simple, intuitive UI there are few surprises in v12 for those who've used LibreELEC or *Kodi* before.

Still, under the hood a number of devices have been changed to 64-bit architectures, including the Pi. This means existing LibreELEC users may have to reinstall *Widevine* if they want to continue watching DRM-protected content like Amazon Prime and Netflix. LXF

VERDICT

DEVELOPER: LibreELEC Team
WEB: https://libreelec.tv
LICENCE: GPL 2.0

FEATURES	**9/10**	**EASE OF USE**	**8/10**
PERFORMANCE	**8/10**	**DOCUMENTATION**	**7/10**

LibreELEC lives up to its promise of "just enough OS" for *Kodi,* offering a lightweight solution for your media needs.

» Rating 8/10

AlmaLinux 9.4

Alma is the Spanish word for soul. **Nate Drake** certainly feels his has been uplifted through this spiritual successor to CentOS.

IN BRIEF

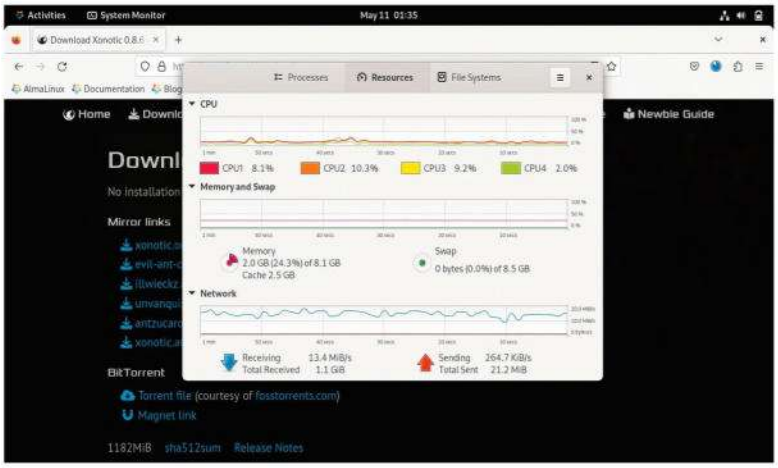
Who says you can't get something for nothing? AlmaLinux offers binary compatibility with RHEL cost-free. Choose the right image to test it in live mode with a variety of desktop environments.

SPECS

CPU: 1GHz
Mem: 1.5GB
HDD: 10GB (20GB recommended)
Builds: x86_64, AArch64, ppc64le, s390x

AlmaLinux was released in 2021 in the wake of Red Hat's announcement that it would discontinue downstream development of CentOS. Like CentOS, AlmaLinux is community-supported and aims for binary compatibility with RHEL.

New versions such as the latest (code-named Seafoam Ocelot) match the release and software versions of RHEL itself. This is done using freely available open source code via the AlmaLinux Build System. This is worthy of an article in its own right; the AlmaLinux Foundation has a 23 minute YouTube video on it works.



AlmaLinux has updated drivers for older gear but needed 2GB of RAM to download a 1.2GB file.

Like Red Hat, there's a number of AlmaLinux images available. We chose from one of the many download mirrors and saw there were three main ISOs available: a boot image of around 1GB, a minimal image of around 2GB, and a 10GB DVD image.

We used one of the offered download mirrors to set up AlmaLinux 9.4 in a virtual machine, as well as manually installing a desktop environment. Spare yourself this and test the OS in a live environment by downloading an AlmaLinux image from **https://repo.almalinux.org/almalinux/9/live/x86_64/**.

This not only allows you to try the OS without installing but offers a variety of desktop environments including Gnome Mini (core environment with a *Firefox* browser), Gnome proper, KDE, Xfce, MATE and more.

On the plus side, using our chosen download mirror allowed us to go through the install process, which is virtually identical to RHEL. You must specify the partitions (automatic partitioning and disk encryption are supported) and create a user account. You can also optionally set a root password, which we recommend as our created user account wasn't a member of the sudo group on first boot.

The bundled *Firefox* browser defaults to the AlmaLinux homepage and includes useful bookmarks such as Documentation, which points to the comprehensive AlmaLinux wiki. This contains easy-to-follow step-by-step instructions for setup, as well as comprehensive release notes for Seafoam Ocelot.

It was here we learned that AlmaLinux 9.4 comes with extended hardware support. The developers claim this was done in response to requests by community members running older hardware. A number of device drivers have been modified to re-add PCI IDs for hardware disabled in upstream.

The wiki also lists other differences between AlmaLinux and the latest version of RHEL by detailing packages that have been removed, though these are all logical, such as the absent **redhat-release-eula**.

Nevertheless, we were eager to test the claims of full RHEL binary compatibility, so fired up a terminal to download Red Hat's *Subscription Manager,* which downloaded in moments and worked out of the box.

We had more of a lacklustre experience using *Gnome Software* to browse for apps, as certain categories only had a handful of available programs, and the Play category was empty. This is a shame, as despite the DVD image having a hefty install footprint of around 61GB, there are few pre-installed third-party apps. Nevertheless, the Updates section worked and brought the OS up to speed within a few moments.

System Monitor reveals that AlmaLinux 9.4 is quite forgiving at rest. With no running programs besides the monitor, CPU usage hovers around 1% and RAM at 1.5GB, in line with the documented requirements.

This spiked to 8% and 2GB when using *Firefox* to download a ZIP of Nate's favourite Linux game *Xonotic.* This makes us wonder if AlmaLinux would play nicely on older hardware, despite the revamped drivers. LXF

VERDICT

DEVELOPER: The AlmaLinux OS Foundation
WEB: http://almalinux.org
LICENCE: GPL 2.0 and others

FEATURES	**7/10**	**EASE OF USE**	**8/10**
PERFORMANCE	**8/10**	**DOCUMENTATION**	**9/10**

Excellent documentation and setup is fairly simple. Install footprint is heavy but the system performs relatively well.

» Rating 8/10

SparkyLinux 2024.05

Nate Drake blinks and almost misses this zippy Debian-based distro, so light on resources it runs perfectly on ancient hardware.

IN BRIEF

SparkyLinux offers the ultimate in customisability, given the number of packages and desktop environments. The pre-installed apps are well thought out, setup is lightning fast, and system requirements are very low..

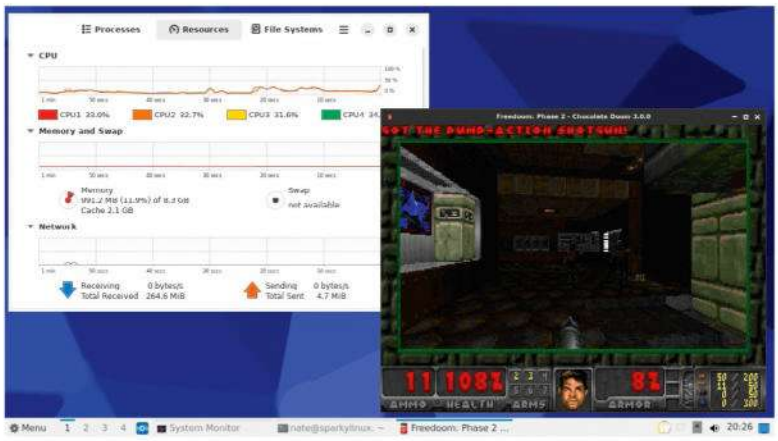
SPECS

CPU: i686 or AMD64, Pentium 4/ AMD Athlon
Mem: 256MB
HDD: 10GB
Builds: IA-32, x86-64, ARM

SparkyLinux is based on Debian. There are two main flavours, one of which is based on the latest stable release of Debian (in this case Debian 12). The other is based on the unstable branch of Debian and is the focus of this review.

The main website describes this version as for "more advanced users, who aren't afraid of a little less stable version of applications".

As our most unstable writer by far, Nate was well suited to put SparkyLinux through its paces. But he hit a snag when downloading ISOs from the main website, due to his browser blocking the files as insecure. On closer inspection, it seems while **sparkylinux.org** has a valid SSL certificate, file downloads take place over regular HTTP. So, we recommend using the provided md5/SHA checksums to verify download integrity. You can also choose the SourceForge mirror, which uses a secure connection.



SparkyLinux is designed to be lightweight. It consumes few system resources, even when performing system upgrades or running graphics-intensive applications.

Customisability seems to lie at the heart of the Sparky ethos, as ISOs of around 2GB are available with the LXQt, MATE, Xfce and KDE desktop environments. You can also install further packages, codecs and desktop environments post-install via *Sparky APTus*.

There are Multimedia and GameOver spins, as well as a CLI edition of SparkyLinux for users to build their own custom desktop and packages. We opted for the Home version running LXQt, which is reflected in the system requirements *(see box, left)*. The Xfce version requires an extra 256MB of RAM to run comfortably.

Given the very forgiving requirements, it's clear that SparkyLinux is designed to be lightweight and we were astonished at the speed at which the live DVD booted into RAM (around 20 seconds).

If you wish to go ahead with setup, it can be handled easily via the integrated *Calamares* installer. The SparkyLinux wiki also references the *Sparky Advanced Installer,* which supports setup on external drives.

No matter which version is installed, you benefit from a respectable selection of pre-installed apps, including *Firefox ESR,* the *LibreOffice* suite, *VLC* media player, *Transmission* and *Thunderbird.*

Users who require more applications can install via the bundled *Synaptic* package manager or *AppCenter.* The release notes for SparkyLinux 2024.05 also state that the *Nala* front-end for **libapt-pkg** has now been removed in favour of plain old *Apt.* You can also carry out updates via the built-in *Sparky-upgrade* utility.

Given this OS's reputation for speed, we decided to run a few tests. Firstly this involved going through the steps in *Calamares*. Despite the notification saying setup would complete in 15-30 minutes, it took less than five. This makes SparkyLinux's setup one of the fastest we've reviewed for a graphical version of Linux.

On first login, we fired up the terminal to install *Gnome System Monitor* from the SparkyLinux repos to find that when only the utility is running, CPU usage hovers around 10%, while the system itself consumed only around 750MB of RAM. Even when running a full system upgrade, or when playing a game of *Freedoom* in windowed mode, RAM usage barely topped 1GB, cementing SparkyLinux's lightweight status.

Special mention should also go to the *YAD*-based welcome window. This actually failed to load in the live version of the DVD. But post-install it appeared immediately on login with helpful links.

The SparkyLinux website does caution that the rolling version can be unstable and the blank welcome window wasn't the only bug we discovered. When trying to extract the *Freedoom* WAD files from a ZIP archive, the Extract Here option in the *PCManFM-Qt* file manager did nothing. LXF

VERDICT

DEVELOPER: SparkyLinux Team
WEB: https://sparkylinux.org
LICENCE: Mainly GPL

FEATURES	9/10	EASE OF USE	9/10
PERFORMANCE	10/10	DOCUMENTATION	8/10

A couple of glitches don't mar its reputation as extremely lightweight and customisable. Ideal for older machines.

» Rating 9/10

Garuda Linux 240428

Nate Drake seldom applies the words 'luxurious' or 'sumptuous' to an OS, but in Garuda's case, he makes an exception.

IN BRIEF

Though spins are available with minimalist window managers, Garuda revels in its flagship edition with rich colours and a tweaked Plasma desktop. You'll need a relatively modern machine to enjoy it properly.

SPECS

CPU: 1GHz
Mem: 4GB (8 GB better)
HDD: 30GB (40GB better)
Builds: x86_64

The Garuda is a large, mythical eagle that features in both Hindu and Buddhist mythology. The name seems fitting for this rolling Arch-based distro, given its rich plumage.

By this we mean Garuda's flagship Dr460nized edition (code name Bird of Prey), which ships with Plasma 6. The developers have put their own spin on this, with neon icons and an awe-inspiring dragon-themed wallpaper. If this isn't to your taste, Garuda offers 47 alternative backgrounds.

In the developers' own words, this edition offers "a dark, blurry and fully immersive Plasma experience". This dark theme is consistent throughout the system; even the OS's custom *Firefox*-based *FireDragon* web browser comes with the dark reader extension.

The welcome screen has a number of home-grown Garuda apps for system maintenance and accessing helpful resources, such as the wiki.

Naturally, such visual splendour comes at a cost, which is why Garuda is also available as a KDE Lite version, which contains a bare minimum of packages to enable you to get started. There are also alternative spins incorporating Gnome, Cinnamon and Xfce.

We focused on the Dr460nized edition of Garuda, but there's also a Gaming version, which is the flagship edition bundled with software like *Steam* and *Lutris*.

While we're on the subject of pre-installed apps, the default edition takes some work to set up for daily use as there's no bundled office suite or email software.

To this end, we fired up the 2.6GB ISO in live mode and launched the *Pacman* front-end *Octopi* from Garuda's dock. This resulted in a database error, but the Garuda forums soon set us straight by saying we needed to run *Garuda-update* before proceeding. We were then able to install *Thunderbird* without issue.

On launching a third-party app for the first time, we saw what the developers describe as its "Mac-like" interface, as the icon appeared alongside others in the lower dock and was customised to match the theme.

The dock also contained a shortcut to the *System Monitor,* which we launched to see how heavy this custom version of Plasma 6 was on system resources. At rest, the system consumed around 1.5GB of RAM, while CPU usage hovered around 3%.

In fairness, the developers do warn against booting up the OS in a VM as we did, because it can cause compatibility issues. The Garuda Linux wiki also explains that system requirements are high mainly due to the requirements of the *Calamares* installer, as well as Arch's own performance tweaks. The wiki also notes that idle RAM usage is higher relative to other operating systems due to using zram, citing the old adage 'unused RAM is wasted RAM'.

We decided to run some more tests by using the *FireDragon* browser to download and run a simple game. On searching for the website, we discovered that searches are handled by the open source, privacy-friendly Searx metasearch engine.

The *Dolphin* file manager opened effortlessly and extracted the 1GB ZIP archive in under a second. During gameplay of the first level of the FPS *Xonotic,* the system used around 5GB of RAM, seemingly confirming the developers' claims that the OS simply uses resources in an efficient way.

Garuda's *Welcome* app is particularly praiseworthy and contains links to key OS-specific apps. These include the *Garuda Assistant,* which can easily perform routine tasks like editing repositories and performing a system update. *Settings Manager* also provides a simple way to carry out tasks like installing language packs. There are also links to Garuda's excellent online documentation, such as the wiki and forums. LXF

VERDICT

DEVELOPER: Garuda Linux Team
WEB: https://garudalinux.org
LICENCE: GPL 3.0

FEATURES	8/10	EASE OF USE	8/10
PERFORMANCE	8/10	DOCUMENTATION	9/10

Garuda is a visual feast with a new Plasma 6 desktop and custom Plasmoids. A solid workhorse for gaming or daily use.

» Rating 8/10

Nvidia GeForce Now for Steam Deck

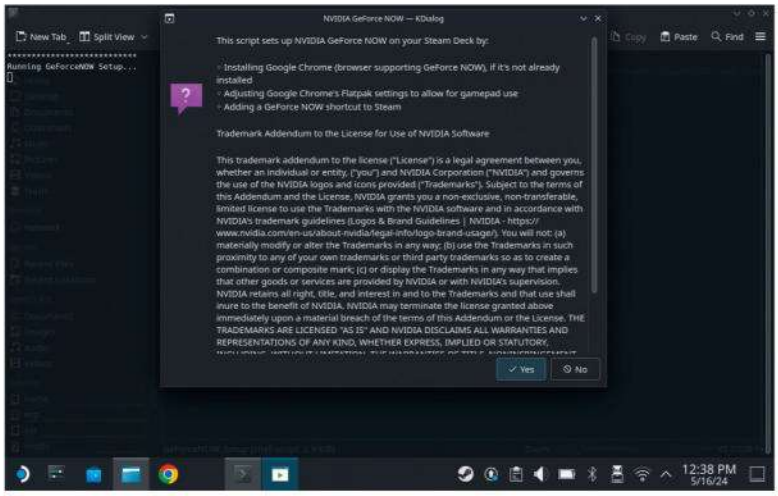
Management is wondering where **Dave Meikleham** and **Alex Wawro** are. Perhaps those noises from the server dungeon aren't those of hard work...

IN BRIEF

Nvidia's own game streaming service provides top-flight quality and access to a wide selection of your own games (Steam, Epic, GoG) alongside its own gaming library – on paid-for tiers – with titles changing over time. But 4K access and more play time will cost you.

Good news, Steam Deck fans: Nvidia is making it a little easier to use its GeForce Now game streaming service on Valve's Linux-based handheld gaming device.

This is important because, until now, getting the game streaming service running on a Steam Deck was possible, but tedious because Nvidia's *GeForce Now* Windows 11 app doesn't play well with Steam Deck's SteamOS interface. Instead, many Deck owners use GeForce Now by switching to Desktop Mode, firing up a browser window and streaming their GeForce Now games through it – a useful dodge to get *Fortnite* up and running on the Deck, for example.



Running the Nvidia GeForce Now script under the Steam Deck Desktop mode.

However, completely out of the blue, Nvidia updated its GeForce Now download page with a new Steam Deck (Beta) download link that, when clicked, downloads an applet that automagically puts a link to GeForce Now right in your Steam Deck library, alongside all your other Steam games.

What it's really doing is adding a custom *Google Chrome* shortcut to your Steam Deck library that's automatically configured for gamepad support and optimal game streaming to Valve's handheld. The *GeForce Now* app also now supports browsing via gamepad, which is a big usability upgrade from the days when Steam Deck users had to navigate the app via the handheld's trackpads.

However, it's still far from perfect, which is why it's nice that Nvidia has clearly flagged this download as a beta version of the installer. If you head over to the customer support section (**https://bit.ly/lxf317nvidia**) of Nvidia's website, you can find a detailed help document that outlines multiple ways to install *GeForce Now* on Steam Deck. This includes methods of setting it up yourself if you'd rather not use the new *GeForce Now* installation script.

It's easier than ever to install GeForce Now, but you still have to fiddle with settings to make it easy to navigate with the gamepad or analogue sticks.

Setting up settings

There are some useful tips and tricks from Nvidia in that document that are worth skimming if you're planning on doing this to your Steam Deck, because there are some settings that you should tweak for the optimal GeForce Now experience. Most notably, Nvidia warns that some games require you to enable gamepad input within their in-game settings menus before you can play using the Deck's pad, with *Genshin Impact* getting called out by name as being especially frustrating.

According to Nvidia, "In *Genshin Impact,* the game settings can only be accessed after you have progressed far enough in the game, which we recommend doing on another device with a keyboard and mouse." So, it sounds as though we still have some way to go before streaming GeForce Now games on Steam Deck is as simple or convenient as playing games that are verified to run well on Valve's handheld.

Install GeForce Now

We've been a subscriber to GeForce Now's Ultimate tier since its public inception in 2020, which enables you to play the best PC games with the power of a cloud-based

CREDIT: Bethesda

And here you are logged in and up and running with GeForce Now on the Steam Deck.

RTX 4080 GPU. It can offer sensational experiences, and we've gotten way too excited over how good it feels playing *Cyberpunk 2077* at 60fps on a £200 laptop in the past thanks to Team Green's streaming platform. So, let's walk you through the quick and (reasonably) easy process of installing *GeForce Now* on Steam Deck to see how it performs.

Deck Desktop Mode

Press the Steam button on the left-hand side of your Steam Deck, then scroll down to the Power option. From here, select Switch To Desktop to boot up the handheld's Linux-based OS. As usual, ensure that you're not in Family Mode, and this process is a lot easier if you pair your Deck with a wireless keyboard and mouse – which you can easily do in Gaming Mode by hitting the Steam button, then Settings, and then selecting the Bluetooth option.

Open up your browser of choice in Desktop Mode and download the *GeForce Now* installation file from Team Green's site. Click the Get Started download link under Steam Deck (Beta) in the Gaming Handheld Devices section. After you download the **GeForceNOW_Setup** file, extract it and select Execute to launch the script and press the Yes prompt when it appears to install *GeForce Now* on your Steam Deck.

Gaming Mode

Either use your mouse to navigate back to the Steam Deck's desktop or the handheld device's trackpads. The Gaming Mode shortcut is generally located in the upper-left-hand side of the desktop by default, so click on it to return to Valve's SteamOS.

Hit the Steam button once you're back in Gaming Mode, then select Library. Use the R1 button to scroll to your Non-Steam section, where you'll find and be able to launch your freshly downloaded *Nvidia GeForce Now* app.

Launch time

Once you're in the *GeForce Now* app, you can stream and play a select number of titles you may either own or rent from digital PC platforms, such as Steam, Ubisoft Connect or Xbox Game Pass. Bear in mind that this app is still in beta, so it may glitch on you every now and then.

That wasn't too painful, right? Well, not unless you had to go through the entire process using the Steam Deck's unreliable trackpads. If you've got decent interwebs, GeForce Now is the game streaming service we would recommend above all others, and we've already had some great experiences using the app on our Steam Deck. LXF

» GEFORCE NOW SERVICE

Nvidia GeForce Now has the potential to be one of the most expensive cloud gaming services, or one of the cheapest, depending on what games you own and how frequently you need to play them via the cloud. Rather than offering you a ready-made library, GeForce Now lets you stream PC games that you already own on Steam, GoG, the Epic Games Store and Ubisoft Connect. You can't play every game on those platforms, but you can play more than 1,500 of them, which beats most of the competition by a wide margin.

Nvidia GeForce Now's free tier lets you play for an hour at a time on servers that sometimes have queues. It works well for casual cloud gamers. But you can also pay £10 per month for Priority server access and 1080p resolution with ray tracing — or £20 per month to stream from an Nvidia GeForce RTX 4080 machine, with 4K resolution and eight-hour sessions. In our testing, we found that the free tier's queues were inconvenient, but the performance was good.

Once installed, GeForce Now will be waiting for you in your non-Steam game section.

VERDICT

DEVELOPER: Nvidia
WEB: https://play.geforcenow.com
PRICE: Free (Priority £9.99 per month)

FEATURES	8/10	EASE OF USE	8/10
PERFORMANCE	9/10	VALUE	9/10

Slick, reliable and with outstanding image quality, GeForce Now offers a balanced service for free – within limits.

» Rating 9/10

WE COMPARE TONS OF STUFF SO YOU DON'T HAVE TO!

Roundup

Hyper » Cool Retro Term » Yakuake » Terminator » Terminology

Michael Reed can come across as terminally serious if the only available terminal doesn't support colour.

Terminal emulators

We all need a terminal emulator at some point to access the Linux command line. **Michael Reed** examines five advanced options.

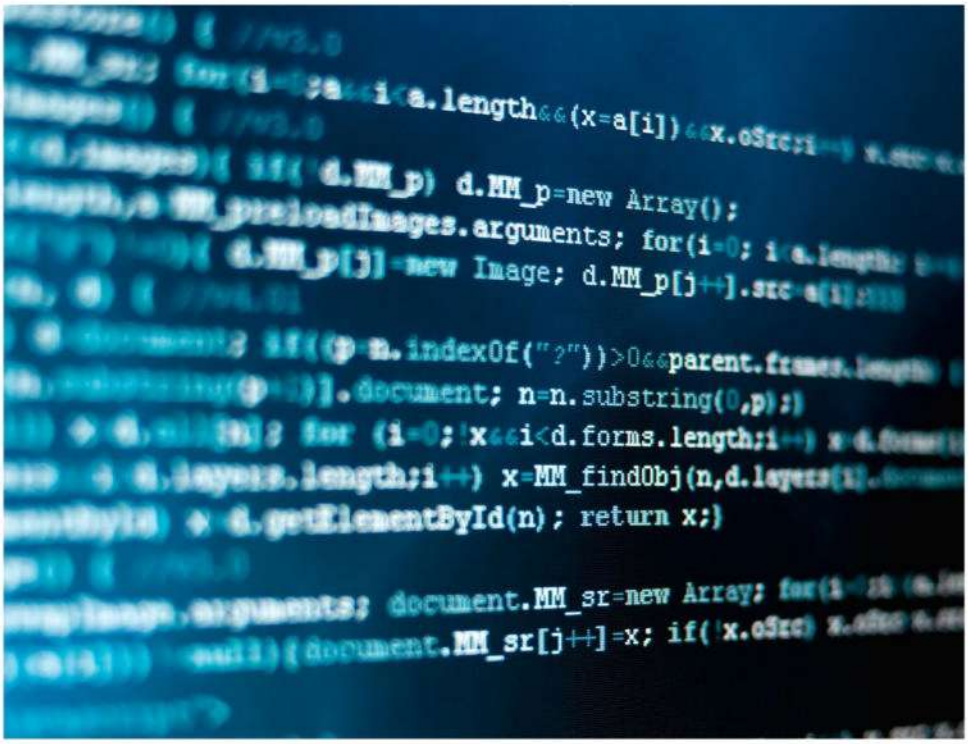
HOW WE TESTED...

The first task was, of course, to install each candidate. We aimed to install the latest stable version in each case. Then it was time to configure each one, particularly in the visual realm. Everyone who uses a terminal emulator has their own take on how it should look, and we messed around with configurations to make ourselves comfortable and ensure that a wide range of options was available.

We lived with each of the terminal emulators while engaged in some text mode activities, such as compiling code and running text mode favourites such as text editor *Nano* and fancy-looking system monitor *htop*.

While we were exploring each terminal emulator, we kept going back to the documentation to make sure that it was comprehensive. There's no point in having fancy features if you can't figure out how to use them.

We've made the odd remark about performance, but generally speaking, all of these candidates offered similar speed in use.



CREDIT: Zokara/Getty Images

Whenever you access the command line from a desktop environment, you need to use a terminal emulator. Every major desktop environment comes with a basic terminal emulator, but there are other choices if you crave better looks and more features.

Yakuake takes inspiration from the terminal in the *Quake* series of games in that it drops down from the top of the screen when you hit a hotkey – an efficient and useful arrangement.

Cool Retro Term cranks up the nostalgia as it can simulate the look of many different types of CRT displays of yore. It looks amazing, but it's also a perfectly good terminal emulator for serious work.

Hyper has a cool-looking minimalist appearance, and it's based upon the *Electron* toolkit and has a huge scope for customisation for those with the inclination and expertise in CSS, HTML and JavaScript.

Terminology is the terminal emulator of the Enlightenment desktop environment and has a few visual extras compared to most.

Terminator offers extensive screen-splitting features so that you can have multiple terminals within one window. It also has a lot of configuration options and great documentation.

Installation

First things first: you need to obtain the latest version and install it.

Generally, most of these tools were out of date in the Linux Mint repositories. For *Hyper,* we downloaded the DEB archive from the website. This worked fine, and there is also an RPM. Manually installing architecture-specific packages like this can present a problem due to updating, but once it's installed, *Hyper* monitors for updates and automatically installs the latest version for you. This behaviour can be toggled off. There is an AppImage file, but that format, which does not install itself, is a different prospect for a system tool such as a terminal emulator. *Hyper* has an option that creates a symlink so it is in the current path, completing the initial setup.

Terminator's main website and documentation don't seem to contain directions for obtaining an up-to-date version. In all fairness, the GitHub page contained information on building the project from the source code or installing via the developer's PPA. We did the latter successfully on Linux Mint, but on non-Debian/Ubuntu-based distros you may have to build from source.

The *Yakuake* website has links to the Snapcraft and Flathub repos, but the Snapcraft package was very out of date, so we went with Flathub. This meant *Yakuake* was installed to the application menus and available from all areas of the system. We didn't run into any problems, due to Flatpak sandboxing.

Our best shot of getting an up-to-date version of *Terminology* seemed to be building from scratch, so that's what we did. Like much of the Enlightenment project, the build process involves using *Ninja* and *CMake.* We'd rate it as a medium difficulty build as those tools aren't well known, and we had to hunt down some dependencies, but the documentation was helpful in this regard.

```
Run-time dependency intl found: YES
Found pkg-config: /usr/bin/pkg-config (0.29.2)
Run-time dependency edje found: YES 1.26.2
Run-time dependency elementary found: YES 1.26.2
Run-time dependency eina found: YES 1.26.2
Run-time dependency eet found: YES 1.26.2
Run-time dependency evas found: YES 1.26.2
Run-time dependency ecore found: YES 1.26.2
Run-time dependency ecore-evas found: YES 1.26.2
Run-time dependency ecore-file found: YES 1.26.2
Run-time dependency emotion found: YES 1.26.2
Run-time dependency ecore-input found: YES 1.26.2
Run-time dependency ecore-imf found: YES 1.26.2
Run-time dependency ecore-imf-evas found: YES 1.26.2
Run-time dependency ecore-ipc found: YES 1.26.2
Run-time dependency efreet found: YES 1.26.2
```

For Terminology, we built it from scratch on our Linux Mint system. It wasn't a completely simple build, but the instructions were helpful.

Cool Retro Term hasn't been updated in a while, which means we could install it in the standard way using `sudo apt install` on a Linux Mint machine. The GitHub page had a list of source code releases for those who would prefer to build from scratch.

VERDICT

HYPER	8/10	TERMINATOR	6/10
COOL RETRO TERM	7/10	TERMINOLOGY	6/10
YAKUAKE	7/10		

All are open source and buildable from source code. Hyper has a reasonable set of installation options and an internal update checker.

Configuration

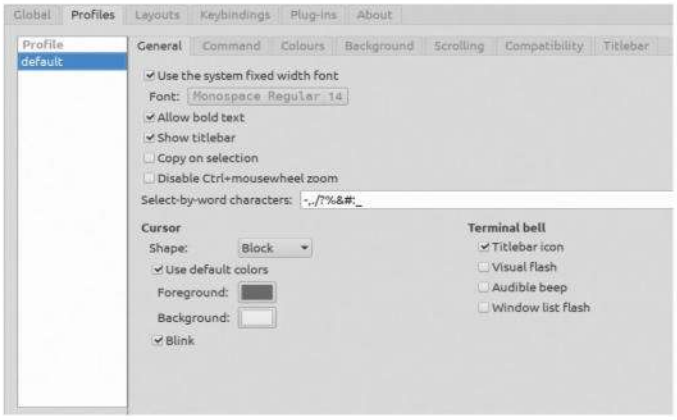
Options to make the experience feel like a personalised home base.

Hyper uses a text file configuration, which opens in the system text editor from a menu entry. It monitors the configuration file and gives instant updates on screen. This gives it some of the inherent advantages of a GUI configuration page mixed in with those of a text configuration file system. The configuration file itself is easy to navigate.

Ninety per cent of *Cool Retro Term's* configuration options control the appearance options. This means that there isn't a huge amount of fine control over things such as running external scripts and setting up hotkey commands.

The *Terminator* configuration is a multi-tabbed dialog, and some of the sections have their own tabs. There are a lot of options for behaviour and appearance, and they are well organised, but they don't take effect until you press the close button on the configuration dialog. The options are wide ranging and cover the standard features you would expect, such as those affecting appearance and how the mouse and keyboard are configured.

The *Terminology* configuration dialog has a strong bias towards cosmetic settings. Like most of its user interface, the configuration is a bit unusual in how it works, using a panel rather than a system dialog. Changes are reflected immediately, and there is a Temporary toggle if you just want to try things out.



Terminator has a lot of well-organised options. For example, within the Profiles tab, there are seven further tabs giving a great number of customisation options.

When you get into the *Yakuake* dialog, you'll find a set of configuration options that cover the basics, without quite getting into the levels of detail that *Hyper* or *Terminator* allow. In its favour, the configuration dialog is searchable, which we are always glad to see.

VERDICT

HYPER	9/10	TERMINATOR	9/10
COOL RETRO TERM	5/10	TERMINOLOGY	8/10
YAKUAKE	7/10		

Hyper is made to be reconfigured, but that does mean digging into its configuration file. Terminator has a lot of options.

User interface and appearance

Eye candy, functionality and ease of use.

Some Linux users like a black box with no adornments whatsoever, but the majority of us appreciate some user interface elements, particularly if they make the less common operations easy to discover.

We like customisability in this area of computer use, and this should consist of facilities to alter the broad strokes such as the colour theme and the font size, along with, ideally, the ability to alter at least the foreground and background colours, and being able to redefine every colour from within the configuration interface. An extra point if visual tweaks are immediately reflected within the terminal.

The cosmetic aspect of a terminal is largely down to personal preference. Ideally, there should be a mixture of selectable presets and specific options for colour elements.

We've given higher marks to terminal emulators that offer good graphical user interface elements as well as being usable from a purely keyboard-controlled realm.

Hyper 8/10

Hyper has black, hackerish looks. It's not immediately obvious, but the equivalent of a pull-down menu lives in the little upper-left icon in the title bar. The minimalism of the user interface is obviously a deliberate design choice, but we might have liked to see icons for things such as keeping the terminal window on top, rather than having to navigate to it in the main menu. The pop-up menu is nicely laid out, with the essentials such as cut and paste and horizontal and vertical split options.

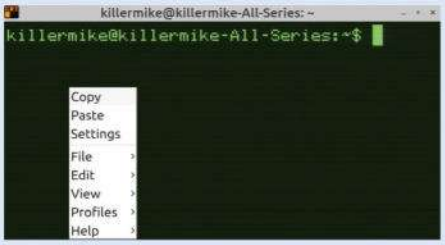
If you invoke the search mode, a white text entry box appears in the top-right corner, making us feel that sometimes there can be too much contrast within a colour scheme. Every aspect of the colour scheme can be customised from within the configuration file, but there is no theming selection from presets – a mark against *Hyper* compared to some of the competition.

Cool Retro Term 9/10

It's a perfectly capable terminal emulator, generally, but the visual side of things is the main draw of *Cool Retro Term*. To us, its defaults felt overstated for practical use as the various effects made text difficult to see. It's a bit inaccurate, too, unless your old CRT monitor had something wrong with it that caused the picture to flicker and wobble all the time.

For actual use, we recommend visiting the configuration dialog to dial down the effects while adding a bit of warmth, fuzzy edges and movement to the picture to give a retro feel to the proceedings. In addition to these controls, there are some preset themes and direct foreground and background colour settings. All settings are reflected immediately, thankfully, given the visual focus. We liked being able to enable a traditional pull-down menu.

The post-processing may seem like a gimmick, but you may grow to love the retro look, if you turn the settings down.



For the beginners

Suitability for newbies or those who don't relish visiting the command line.

Hyper has a plain default setup, without a toolbar or a pull-down menu. Its configuration is carried out through a text file, and even though *Hyper* automatically senses changes to that file, this makes it less suitable for beginners. Most of the terminal itself works in a fairly conventional way. For example, you can highlight text and right-click to copy it.

As the *Yakuake* window drops down from above, it doesn't work in quite the same way as other terminal apps, but once this is understood, it's a perfectly user-friendly terminal emulator. The arrangement is highly ergonomic in use, eliminating manual window management entirely, particularly when trying to copy instructions from a website tutorial.

Terminator can be used as a normal Linux terminal emulator, but the only icon on the interface relates to the group feature, a feature no beginner is likely to need. The splits are also unlikely to be useful to a Linux newcomer, and if accidentally invoked, could lead to confusion. Its huge number of configuration options are likely to be an unnecessary distraction to newbies.

Terminology doesn't quite adhere to the normal user interface conventions for a terminal emulator because there is no pop-up menu as such. However, it doesn't take much to learn its quirky ways, and the general user will probably appreciate its cosmetic options.

Cool Retro Term shouldn't present any huge difficulties for a newcomer because it works in much the same way as any other terminal emulator, apart from the graphical post-processing. Also in its favour is the fact that it has a standard pull-down menu that can be enabled.

VERDICT

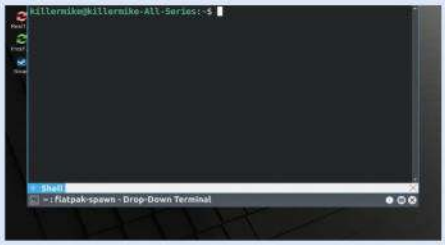
HYPER	6/10	TERMINATOR	6/10
COOL RETRO TERM	8/10	TERMINOLOGY	8/10
YAKUAKE	7/10		

Cool Retro Term is a good basic terminal emulator and adds some fun to working with the command-line interface.

Yakuake 8/10

Yakuake is the odd one out because it slides out of the top of the screen when a hotkey is pressed. This is more than a gimmick because it means that the terminal is always close to hand, and it enables you to set up some useful splits of the screen with ease, much more convenient than trying to juxtapose the normal rectangular window that most terminal emulators use.

Once it's on screen, the default palette is a mellow grey and white, with use of pastels for colour highlights within the window. Along the bottom, there is a status bar that includes icons for the stay-on-top toggle and the main menu. This menu has the look of typical KDE applications. Handily, the status area has the + icon for creating extra tabs. The options are a bit limited for configuring a custom colour setup, but it includes a shop-style theme browser, with dozens of themes downloadable.

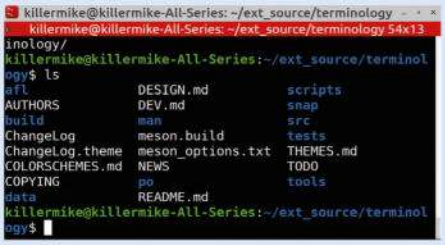


Terminator 7/10

Terminator is heavily orientated towards quickly and easily splitting the main window horizontally or vertically, and this is done using the right-click context menu or through a key combination. Layouts can be saved for future recall.

Each terminal window pane has a small title bar with a single drop-down icon menu for controlling broadcasting, which is *Terminator's* feature for sending the same commands to multiple terminals. It's an icon bar with one icon. It's a shame that more isn't made of this area to give quick access to other features, such as creating new tabs and splitting the window.

Terminator's default palette is grey on black and it uses a scheme of darkening deselected panes, making everything look a bit muted. It's an easy fix to select the white-on-black colour scheme. There are several useful themes and it's possible to redefine individual colours to get things looking exactly how you want them.



Terminology 8/10

Terminology is the terminal of the Enlightenment desktop environment, so it comes as no surprise that the interface is quirky with a few flashy extras. Right-click in the window and a control panel slides in. This is where you can create new tabs, split the window horizontally or vertically, or access the settings dialog.

The miniview can also be toggled from this control panel. This scroll bar with a miniaturised graphical view of the overall content is similar to the one in *Visual Studio Code*. There are features for viewing image and movie previews inline.

Colour selection uses colour schemes. There are a lot to choose from, which is good, but editing them means manually editing the configuration files, which is less good. All of the cosmetic features that you can edit are immediately carried out upon selection – a point in its favour.

It's efficient and looks good, with ample options for visual customisation.

For the experts

How suitable is it for advanced users such as programmers or sysadmins?

Hyper's reliance on a text file for configuration will appeal to many more experienced Linux users, and it's a boon that *Hyper* both scans for changes to the file and can open the file via the main menu. In theory, you could customise how this terminal operates if you're familiar with CSS, HTML and JavaScript. What percentage of users would actually do that is hard to say. But you could, in theory, create a highly customised system that works exactly how you want it to. The documentation is aimed at programmers who would like to extend and customise *Hyper.*

Terminator emphasises the creation of horizontal and vertical splits in the terminal window, meaning that you can have more than one terminal at once in a tiled arrangement. As well as creating these arrangements, it's possible to store them and recall them for later use. That's certainly a power user feature. The groups feature, which enables you to output to more than one terminal at once, along with the plugins system, are also specialist features.

Expert users will appreciate *Terminology's* full set of configuration options and split window feature.

It's nice looking, but *Cool Retro Term* doesn't have a lot of detailed configuration features that aren't connected to eye candy. It lacks split-screen views and detailed documentation.

Yakuake is in a similar boat, but in its favour, it's a highly practical system for working through web-based tutorials. Exploring its menu system uncovers a few extra features, such as being able to save the contents of the buffer and set up multiple configuration profiles.

VERDICT

HYPER	9/10	TERMINATOR	9/10
COOL RETRO TERM	5/10	TERMINOLOGY	7/10
YAKUAKE	6/10		

Hyper is aimed at expert users and coders, and Terminology also offers a lot of configuration options and expert features.

Extensions and extensibility

Adding extra features and your own unique functions.

If you have the necessary scripting skills, you could just about build your very own terminal emulator program by using *Hyper* as your base and then adding various features to it – it's that flexible. Thankfully, the documentation covers every aspect of working with the codebase and configuration files.

Hyper also has about 25 ready-made plugins listed on the website. Most of these carry out simple functions, such as answering Y to a yes or no prompt, or altering the way the user interface functions, rather than adding a great deal of extra functionality. They are well documented on the website, with some of the plugins featuring an animated example of what the plugin does, and they could also serve as examples for creating your own plugins.

Plugins are the main way of extending *Terminator's* functionality. It has about a dozen plugins that can be enabled in the configuration dialog. Most of the plugins are fairly simple, such as an activity notification one and one to add custom commands to the context menu. The website details exactly what the plugins do and also links to about a dozen third-party *Terminator* plugins on GitHub. The plugins themselves are written in Python and the official documentation explains how to create your own plugins.

```
"""
Terminator plugin to open a file using a chosen editor.

Author: michele.silva@gmail.com
License: GPLv2
Site: https://github.com/mchelem/terminator-editor-plugin
"""
import inspect
import os
import re
import shlex
import subprocess
from terminatorlib import plugin, config

AVAILABLE = ['EditorPlugin']
```

Editing the code for a Terminator extension on GitHub in Visual Studio Code. The extensions themselves are written in Python and there are lots of examples.

Cool Retro Term, Yakuake and *Terminology* don't have much in the way of adding plugins or extension facilities. They don't quite get zero points in this category because you could change the shell that they use or add your own scripts. As they are open source projects, you could customise how they work or add new features if you are a programmer.

VERDICT

HYPER	9/10	TERMINATOR	7/10
COOL RETRO TERM	3/10	TERMINOLOGY	3/10
YAKUAKE	3/10		

Hyper is the leader when it comes to extensibility; however, Terminator also has a plugin system.

Documentation

How to use a given terminal emulator and how to customise it.

There are two main sources of documentation for *Terminology*. The long, single page of the official website is worth a read, as it gives an overview of the features. This is helpful as many parts of this system are unconventional in how they work. The GitHub page gives information on building *Terminology,* in addition to an overview of basic controls. There is also documentation within the GitHub page covering areas such as customising colour scheme files. Overall, the documentation covers most areas adequately for a project that is orientated towards users who want a good-looking terminal emulator rather than those who want to customise every aspect of how it works.

Terminator uses **https://readthedocs.com** for documentation. It is written in a nice chatty style and has the feel of a user manual that covers every aspect of this terminal. We found some of the developer's own testing of the memory use and overall efficiency (in the FAQ) interesting. This comprehensiveness is appropriate for a project aimed at users who want to get a bit technical.

Yakuake has little in the way of documentation. There is an IRC channel with a few people in it. The GitHub page has instructions for building *Yakuake*. It's just as well that *Yakuake* is easy to use.

In a similar way, the *Cool Retro Term* GitHub page covers building the project, but you are obviously expected to explore the user interface to figure things out on your own.

Let's walk through its code.
First, we intercept the Redux action `SESSION_ADD_DATA`. You can find the full list of actions in the repository.

```
exports.middleware = (store) => (next) => (action) => {
  if ('SESSION_ADD_DATA' === action.type) {
    const { data } = action;
    if (/bash: wow: command not found/.test(data)) {
      store.dispatch({
        type: 'WOW_MODE_TOGGLE'
      });
    } else {
      next(action);
    }
  } else {
    next(action);
  }
};
```

The Hyper website goes into quite a lot of detail when it comes to customising and extending it. This section shows how to add some animation features using React.

The *Hyper* website is a long page of information about the project. *Hyper* uses a text file for configuration, and this page covers every configuration option. It also covers things like the extension API and customising key commands. A single, long page is an odd way of organising the information, but as it covers every aspect of some quite complex software, we can't really fault it too much.

VERDICT

HYPER	9/10	TERMINATOR	9/10
COOL RETRO TERM	5/10	TERMINOLOGY	8/10
YAKUAKE	5/10		

Terminator's web documentation is a comprehensive user manual. Hyper's documentation covers every area of the software.

The verdict

Terminal emulators

We'll choose *Terminator* as the best all-round option for an enhanced terminal emulator, even though all of the candidates are good enough to recommend if you prefer the way they look and work.

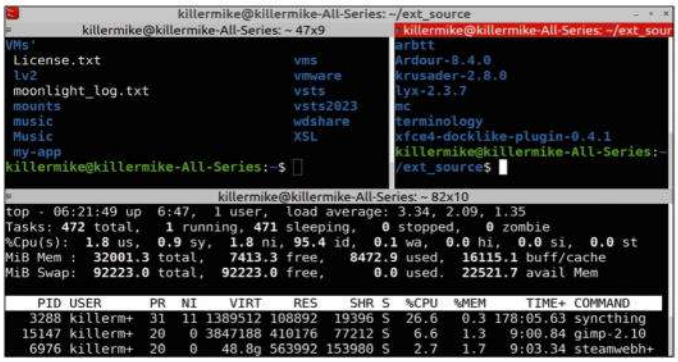
Terminator's most prominent feature is that it can split the main window horizontally and vertically, and save the layouts for regular use. This tiled approach is useful in many situations, but some of the other options also offer splits. It is highly configurable, and if you have an unusual requirement in an area such as a custom keyboard function, *Terminator* can probably do it.

Hyper's configuration system is text-based, but it can detect any changes to the configuration file without requiring a restart. That said, GUI configuration dialogs do offer a lot, and it lacks niceties such as a set of preset themes. It's based on the *Electron* toolkit and programmed in JavaScript, and if you're already actively involved in that type of development, this might make it an all-the-more attractive choice. We found that it was slightly slower to launch than the other options, but not massively so. Still, we wouldn't recommend it for low-resource situations such as old computers.

Yakuake's main gimmick, that it pops down from the top of the screen, is actually useful. It's not just that *Yakuake* is always a hotkey away. There are many situations in which you can line up the text from a web page for a handy split-screen setup when you're following instructions. Beyond that, it's fast and has the normal GUI features you'd expect, even though the main menu is a little bit hidden behind an icon that you might not spot upon first use. It works perfectly well, but we gave it a slightly lower mark as it hasn't been updated in a while.

We recommend toning *Cool Retro Term's* visual settings down so it's still got a bit of retro glowing charm and soft edges, rather than looking like something you'd find in *Fallout*. Once you do, it's a perfectly usable general-purpose terminal emulator, but it lacks some of the more technical features of other options, such as screen splits.

Terminology is aimed at the user who wants a lot of control over how their terminal emulator looks. The flip side is that it lacks much scope for extension by either installing plugins or creating your own.



1st Terminator 8/10

Web: **https://gnome-terminator.org**
Licence: **GPL 2** **Version:** **2.1.3**
Handy vertical and horizontal splits. Good configuration dialogs and options.

2nd Hyper 8/10

Web: **https://hyper.is**
Licence: **MIT** **Version:** **3.4.1**
Highly configurable and extendable. Covers the advanced-level features.

3rd Yakuake 7/10

Web: **https://apps.kde.org/en-gb/yakuake**
Licence: **GPL 2** **Version:** **24.02.2**
Clever interface design and nice looks. Good all-round features.

4th Cool Retro Term 7/10

Web: **https://github.com/Swordfish90/cool-retro-term**
Licence: **GPL 2/3** **Version:** **1.2.0**
Charming retro looks. Full basic features. Real-time configuration.

5th Terminology 7/10

Web: **www.enlightenment.org/about-terminology.md**
Licence: **BSD 2-Clause** **Version:** **1.13.0**
Great looking and easy to use. Interesting inline thumbnail previews.

» ALSO CONSIDER

Every desktop environment comes with a terminal emulator of some kind. For example, the *Gnome Terminal* comes with Gnome, and it features some integration with that desktop environment. Much the same can be said for *Konsole,* the KDE terminal emulator. They tend to provide a perfectly usable environment for basic tasks, and these native, default terminal emulators might be all you need, particularly if you're only an occasional CLI user.

By default, terminal emulator *Alacrity* doesn't have any menus or the scroll bar enabled, yet it uses OpenGL for mega text-rendering performance. It's one for the hardcore text-mode fanatics, and we've kept away from the super-bare-bones terminal emulators on this occasion.

Kitty nearly made the list. It has a lot of configuration scope for serious users, but it also has some nice graphical features and a useful status area at the top of the window. LXF

REPAIR IT WITH LINUX

Don't consign your old machines to the scrap heap just yet – **Jonni Bidwell** reckons they can still be put to work.

We've all got a PC or seven lurking in the basement gathering dust. And if LXF Towers is anything to go by, there may be enough spare parts to build several more. Yet the demands of modern operating systems, particularly those of Microsoft, make it seem that this old hardware is obsolete. Well, we're here to tell you this is not true. Big-name Linux distributions might require hardware that's less than a dozen years old, but there are plenty of distros that run quite happily on machines approaching 20.

If you have a 32-bit machine, for example, you can't run the latest version of Ubuntu, but you can run Debian 12 (released in 2023). Or anything based on it, such as the fantastic Linux Mint Debian Edition (LMDE), AntiX or MX Linux. We'll do a deep dive showing the ins and outs of installing LMDE6, as well as installing it on an low-power 14-year old 32-bit netbook.

We've also got essential advice for getting your old hardware upgraded to its maximum potential. And some tips for when that hardware doesn't work quite as it should. With a tiny investment in memory and a solid-state drive, for example, that old Pentium 4 under the stairs could be transformed into a functioning workhorse.

CREDIT: Magictorch

Time and tide

Linux can run on various machines that are plenty old, but first we really ought to manage your expectations.

Many people turn to Linux to breathe new life into their old systems. Consigning ageing machines to the scrap heap (or preferably an ewaste service) may be cathartic, but if it works, it seems a shame not to use it. A machine that is not capable of running Windows 10 *(see box)*, or running it in a usable manner, can in all likelihood run some form of Linux. We'll be clear about this, though: there are some limits, some which depend on how much effort you're willing to invest. Modern Linux can't run on your dusty old 386, for example. Such support was removed in kernel 3.8 over a decade ago. If you really want to build your own Linux distro based on an older kernel, you could, but it's well beyond the scope of this feature.

Mainstream desktop Linux distros (except Debian) have all stopped supporting 32-bit x86 processors (referred to generally as the i386 architecture, or for specific processor generations as i486, i586 and i686). Fedora even stopped supporting earlier 32-bit ARM architectures, so you can't run Fedora 37 or later on early Raspberry Pis. The kernel still supports these architectures, but a kernel without the rest of the OS is pretty useless. Just ask Richard Stallman. And that's not the end of it. Obsolescence exists beyond CPUs. Good luck trying to get Linux running with Vesa Local Bus motherboards, ISA peripheral cards or the like. Even support for humble floppy drives has been demoted to legacy status (USB floppy drives still work, if you really want unreliable, bulky, slow storage media).

Old graphics cards won't work if the drivers haven't been updated for new kernels. AGP support, while still present, is marked as legacy, and there have been calls to remove it altogether. Nvidia stopped supporting its AGP cards on Windows (and through its proprietary Linux driver) in 2017. Laptop graphics are their own kind of special, and legacy laptop graphics may behave erratically. Manufacturers were guilty of tweaking chipsets and specs, so that specific driver tweaks were required. When such hardware stops being tested and no one bothers to file a bug report, these tweaks are lost and that hardware (even though there's ostensibly a driver for its hardware class) stops working.

The delightful operating system that is LMDE Faye can reinvigorate your ageing digital machinery.

Major desktop Linux distros today typically stipulate 4GB of RAM and a 64-bit processor (and some form of graphical hardware acceleration for modern desktop environments). This means if you have pretty much any PC less than 15 years old, you can get Linux running on it with little effort or investment. If you're prepared to go more off-piste, you'll find distros that run out of the box on Pentium Pro CPUs (introduced in 1995 and bearing SSE instructions). And if you put in the work, you might even get that 486 booting to a terminal.

» WINDOWS END-OF-LIFE CARE

Microsoft got a lot of flak (and rightly so) for forcefully upgrading Windows 7 and 8 machines to Windows 10. Its argument was that such PCs were a risk to the internet, as for years their OSes had been unsupported. It's a fair point (see 2017's WannaCry ransomware outbreak), poorly executed. Windows 10 had relatively low (for the time) system requirements, namely 1GB of RAM and a 1GHz CPU, and getting its userbase migrated to it (or crippling their machines with failed upgrades) meant less of a headache for Microsoft.

At one time, Windows 10 was going to be the last release of Windows, and a rolling, always maintained utopia. Until Windows 11 was announced. And now Windows 10 support is slated to end in 2028. Windows 11, apart from not being to everyone's taste, has much more demanding system requirements, including TPM 2.0 chips, UEFI firmware and 4GB of RAM. It is possible (with some registry-editing fu) to circumvent these, but then you are running an unsupported configuration, and thus should expect random breakage. So, we imagine a wave of users will be looking to migrate to Linux, as has happened with pretty much every Windows release since Windows 98.

Hardware refresh

Fifty British pounds (or local equivalent) and half an hour on an auction site might give your PC the shot in the arm it craves.

Whatever machine you're trying to resurrect or rejuvenate, it's always worth getting as much RAM into it as possible. Retailers won't stock this any more, but you'll find plenty of DDR1 or SDRAM sticks on auction sites for cheap. Just make sure to check your motherboard's manual to find which speeds and stick combinations are required and whether you need ECC memory (unlikely). The model number for the motherboard should be visible in the BIOS somewhere, so type that into DuckDuckGo and you'll find a manual in no time. Later old hardware is much less fussy than old-old hardware about mixing and matching memory capacities. So, if your machine is old-old, aim to put identical sticks in all the slots. Or in a pair of the slots (usually the two nearest the CPU or two of the same colour) if that maxes out capacity. You can, generally, mix speeds, but the faster modules will only run at the speed of the slower ones.

If you have less than 4GB of RAM, all this PAE talk should not concern you.

Speaking of memory, now's a good time to dispel the myth that 32-bit hardware cannot use more than 4GB of RAM. This is not true. At least, it is not true for all 32-bit hardware. Windows XP did odd memory management, so that only about 3GB of RAM could actually be addressed. Early 32-bit hardware rarely supported anything close to 4GB of memory (your typical 386 wouldn't have had more than 4MB), with the first consumer-grade machines sporting such opulence not appearing until the early 2000s. But Linux didn't support it due to bus restrictions, rather than any logical limitations. In fact, if 1GB memory modules had been widely available (for less than the GDP of a small country) in 1995, and your motherboard could support more than four of them, you could happily make use of all that memory.

House of PAE-n

That year, '95, saw the introduction of the Portable Address Extension (PAE) on Intel's Pentium Pro and AMD's Athlon processors. A little maths will reassure you that you need 32 bits to uniquely address every byte amongst four gigabytes: 10 for each byte in a kilobyte, 20 for the kilobytes in a gigabyte, and two more for your four GB. Technically, we should use the kibibyte, mibibyte, gibibyte units here (as we're working with powers of two and not powers of ten), but we think they look silly. However, if your machine has no registers wider than 32 bits, there's nothing to stop it using two (or, more correctly, one and part of another one) such registers to address more. This is what the PAE CPU extension standardised. Obviously, chipsets and buses would need to understand this addressing, too, and such hardware didn't come about until the early noughties. PAE allows for a 36-bit address space,

» UPGRADING CPU AND GRAPHICS

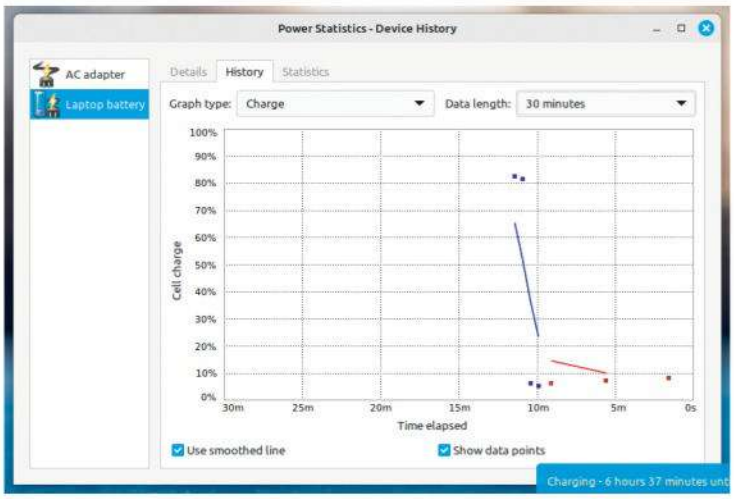
You can upgrade your CPU, too, but the benefits are often overestimated (unless your motherboard supports much faster CPUs). Check the manual to see what's supported. It may be that a BIOS upgrade is required in order to support newer processors. This has to be done before you put the new CPU in, and for old systems generally involves booting to DOS (or a DOS-like OS such as FreeDOS) to run the manufacturer's update utility.

Be very careful when downloading new BIOSes; motherboards often come in lots of minor revisions. You might need to look for an engraving on the board to see which variety yours is. In some cases (such as where USB booting isn't supported), you may need to create a bootable FreeDOS CD (or DVD) with the required BIOS on it. This can be involved, to say the least. A few manufacturers allow BIOS updates using a Windows program, so you might want to hold fire on nuking your Windows XP/7 partition.

If you have an old desktop machine, particularly one with integrated graphics, you might want to hunt for a compatible discrete graphics card. Integrated graphics of old (with the exception of most Intel stuff) have always been problematic. So, you might save yourself a few headaches and boost your retro-gaming abilities by finding a fast-for-the-time card. If the motherboard has an AGP slot, though, it might cause more problems *(see discussion on previous page).*

enabling up to 64GB of memory (you can do the maths this time). The caveat is that each application (or each process of an app) can only use 4GB.

If your machine doesn't support PAE, you can still find flavours of Linux that run quite happily on it. Long-time **LXF** favourite Bodhi Linux's legacy release does so, and suggests only a 500MHz processor and 512MB of RAM (and works on lesser). Ubuntu stopped producing non-PAE kernels in 12.10. There are some odd Pentium-M chips (namely Banias chips, around from 2003-2009) that support PAE but do not advertise the fact. In this case, the `forcepae` kernel option can be used.



If you have an old laptop, like our Eee PC, you should probably invest in a new battery, too.

By now, everyone has pretty much come to terms with newer software running increasingly slowly as hardware ages. This isn't just because of lazy programmers or needless resource-intensive features. In the late '90s, web browsers were pretty much read-only word processors. Now they are Turing-complete behemoths with their own media frameworks, payment APIs and even USB interfaces. Web pages have to be ingested into a memory-hungry DOM (Document Object Model). They have to adapt to all manner of screen/window sizes. JavaScript is compiled by a powerful engine (such as V8 or SpiderMonkey) and fancy sites might even use WebAssembly to run on the machine's native instruction set. Oh, and they need all kinds of expensive precautions against hackers (such as sandboxes).

Anyone using an old machine in the early YouTube days will remember the pain of waiting while *Flash Player* loaded and then misery as the video chugged along. And then the feeling that you'd just wasted minutes of your life on a largely unfulfilling endeavour. That's another example of the gradual phenomenon of hardware behaving slowly as technology evolves. However, sometimes things happen hard and fast. Such as when, in 2013, *Adobe Flash* started requiring SSE2 instructions. It ceased to function at all on hardware without them, bailing out with a serious-sounding 'Illegal instruction' error. This affected Athlon XP and Pentium 3 silicon, hardware that was already at least 10 years old, but it still upset a lot of people. Windows 7 introduced an SSE2 requirement five years later. Thankfully, *Flash* has all but been banished from the web and Win 7 is unsupported. But the curtailing of old hardware support continues. In 2021, *Google Chrome* started requiring SSE3 (the next generation of vector instructions launched nearly two decades ago).

If you've been perusing the latest Linux offerings, you might have noticed -v3 images. This indicates distros that use modern CPU instructions (such as AVX and AVX2, the Advanced Vector eXtensions) that boost computation-heavy and multimedia workloads. There's even a fledgling -v4 industry, for the latest AVX512 extensions. In time, we wouldn't be surprised if the v3 microarchitecture becomes a requirement for all but the most niche distros. This is a feature about old features, so we won't get into this. We just want to stress that developers quite reasonably expect their software won't be running on 20-year-old hardware, and so will cease to support it. Especially if it increases the maintenance burden, such as the 386's quirky memory management.

Hard drivin'

Apart from maxing out your system's memory, the next best thing to do is replace its rotational hard drives with solid-state drives (SSDs). As long as you don't want terabyte capacities, these are ridiculously cheap. And you'll no longer have to listen to your hard drive chugging away as it scrambles to keep up with your OS. If your system is very old, it may have IDE drives with the 80-pin (or 40-pin, if we go way back) ribbon connector. If your motherboard has no SATA ports, you'll have to make do with an adaptor, and the SATA drives won't achieve their stated performance. Obviously, older SATA motherboards won't run newer drives at top speed either, but even the first generation has a throughput of 150 megabits per second – much faster than spinning rust devices. Most hardware from the last 15 years supports SATA3 and 600Mb/s, which should give you a nice bit of I/O boosting.

It's generally overlooked, but if you have an old system, you might want to replace the PSU. Old hardware is not known for its efficiency at the best of times (despite CPU frequency scaling having been around for two decades). So, a new power supply might stop it swallowing so many coulombs. More importantly, it will obviate the possibility of the old PSU popping. We have seen this happen close up, and can attest that it is a traumatic experience.

Distros such as Bodhi Linux are ideal for old hardware, or new hardware if you want it to fly.

Introducing LMDE

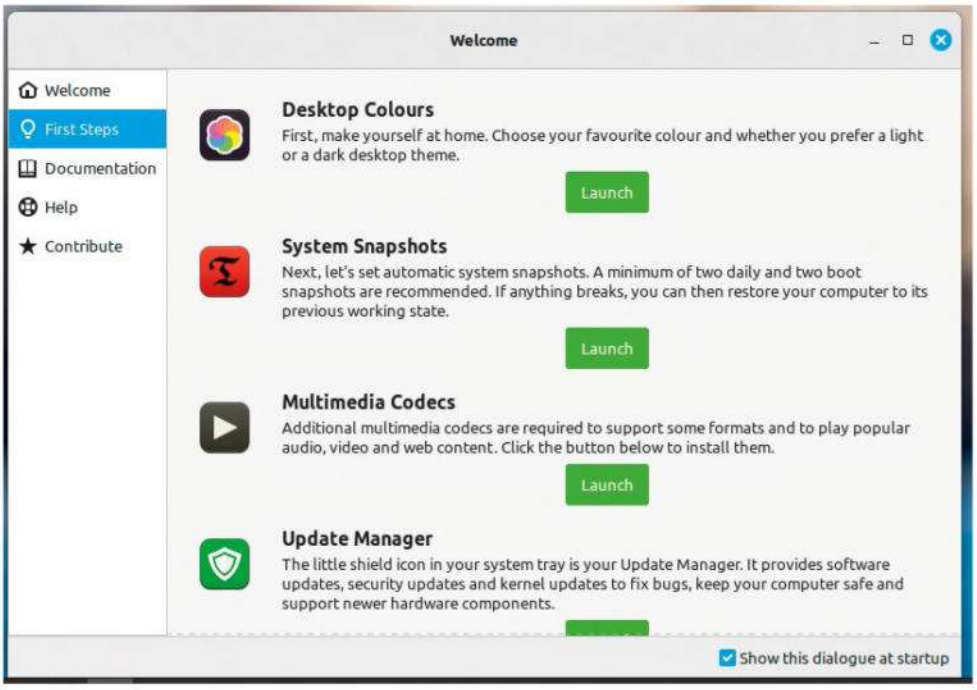
We investigate the easy-to-use but powerful Linux Mint Debian Edition – like regular Mint, but with less Ubuntu, more Debian.

We mentioned earlier that Debian is the only major-league distribution still supporting 32-bit x86 hardware. This is an oversimplification, because we are simple journalists, after all. You are no doubt gasping to point out that several 32-bit versions of Ubuntu can have their support window extended well into the future *(see box, page 38)*. Be that as it may, Debian 12 will be supported until 2028 and, thanks to its configurability, it is a fine distro for your dusty old hardware. There's also a number of lightweight distros based on it, including AntiX, MX Linux and Crunchbangplusplus (aka #!++), a continuation of the popular CrunchBang project. Obviously, there are non-Debian-based distros that still support 32-bit machinery – more on those later.

For this section, we want to concentrate on what, in our opinion, is the most user-friendly distro with 32-bit support: Linux Mint Debian Edition (LMDE). Mint has for a very long time been our go-to recommendation for beginners. And LMDE is, on the surface, just as easy to use. So, if you're refurbishing an old machine for Auntie Ethel to use as a hacking station, LMDE is a fine choice. Of course, LMDE isn't just for beginners *(who are you calling a n00b? – Auntie 'l33t' Ethel)*. If you're not a fan of the slightly Windows-like Cinnamon desktop, you can easily install something else. Most likely this will be something lightweight such as MATE (ideal if you yearn for Gnome 2 stylings), Xfce or even LXQt.

At the time of writing, the latest version is LMDE 6 Faye (based on Debian 12 Bookworm). There will be a new release when Debian 13 appears (some time in 2025). Before you fire up the LMDE 6 install media (which you can find at **www.linuxmint.com/edition.php?id=308**), make sure your system meets the requirements, to wit: 2GB RAM (4GB recommended) and 20GB (100GB recommended) disk space. Per the release notes, the 32-bit LMDE images support older non-PAE CPUs. So, if you want a PAE kernel (and your hardware supports it), you have to add this post install *(see box, below, on how to do this)*. You can boot in Forced PAE mode from the boot menu, but this only affects the live environment, where you're probably not going to need PAE support.

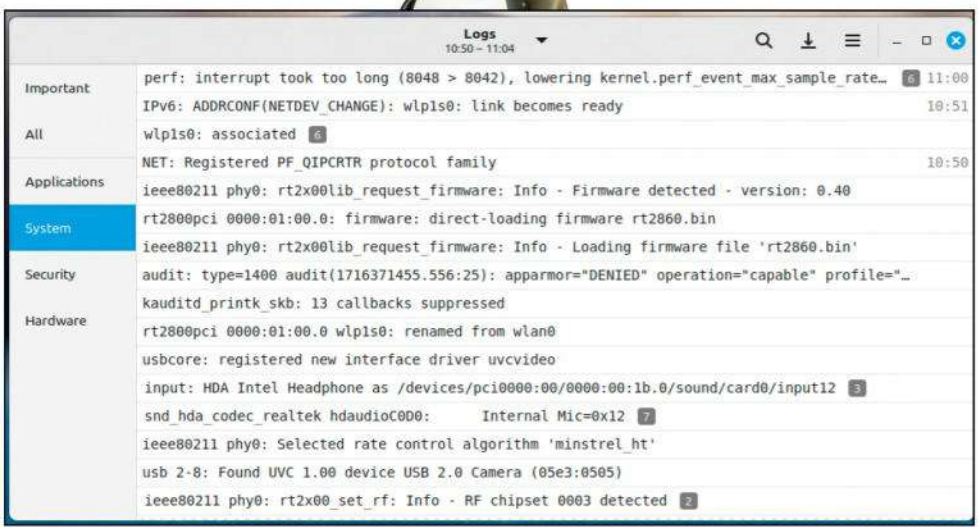
As mentioned earlier, older machines may not support USB booting. Fortunately, LMDE 6 easily fits on a DVD-R, so if you still have an optical drive, booting LMDE will be no problem. Some BIOSes have an option for USB Booting, but on trying it you may find it doesn't boot USB sticks (or only does so if they're in one particular port). Don't give up hope. Our Eee 901 (as covered extensively by Mike 'MikeOS' Sanders in **LXF106-LXF109**) exhibited this behaviour. What worked for us was choosing the Hard Drive boot option instead and messing with the drive order (USB sticks presumably being treated differently from USB hard drives). Hopefully, you manage to at least boot LMDE. If you get really stuck (or are using very old technology that doesn't support booting from optical media or USB), you might want to use something like the *Plop Boot Manager* (**https://plop.at** – hasn't been updated for a decade, but it still works). In an extreme case, you can install this on a floppy, boot from that and then chainload LMDE from USB or DVD.



The welcome screen tells you everything you should do to finish setting up your machine. Disobey it at your peril.

As with any distro, it's worth spending some time exploring LMDE's live environment before you install it. The Cinnamon desktop is fairly lightweight, and it should be easy enough for Windows users to get to grips with. Running the live environment will be much slower than the real thing on old hardware, so don't give up hope if it's a bit laggy. Even booting it may take a long time (thanks in part to USB2/optical media speed limitations, but also processors of old will struggle with the complexities of booting a modern operating system).

We won't walk you through the install process, because you probably have a bit of Linux



LMDE's Logs application can help you identify any system problems or quirks, and save you having to do so from the command line.

experience behind you if you're contemplating dabbling with it on legacy hardware. It's much the same as installing any other distro. We will note that a hidden Expert mode is available in the installer. This enables you to do complex partitioning from the installer. Run:

```
$ sudo live-installer-expert-mode
```

When you get to the partitioning section, you'll find an Expert Mode button. If you have less than 1GB of memory, you might run into issues with the installer. But that's what you get for not heeding the system requirements. You'd honestly be better off looking at another distro. Still, we persevered with running it on the Eee with its 1GB of memory. One interesting conundrum we ran into was that the installer's window was often too tall for the display (again, the requirements state a minimum 1,024x768 and the Eee was 168 pixels shy of this). This is easy to work around, because Cinnamon enables you to move windows from the middle (instead of the title bar) by holding down Alt when you drag. Thus we were able to access the buttons at the bottom.

Again defying the system requirements, we installed LMDE on the Eee's mighty 16GB SSD, a process that took over an hour. And made the device concerningly warm. Nervously we rebooted, and were relieved to see the boot process spring into action. The initial install occupied just shy of 7GB, but with regular use, expect this to balloon.

On first booting LMDE, hopefully you'll not see any glaring error messages and will be quietly impressed at how much more nimble it is once installed. If you do see one of the aforementioned scary-looking error messages, do check the official forums, and in particular the LMDE subforum at **https://forums.linuxmint.com/viewforum.php?f=244**. It's very likely someone has had the same problem as you, so do give them a thorough scour. Especially before posting any requests for help.

Soon after you log in and connect to the internet (assuming you're not stuck with a Wi-Fi device that needs additional firmware or such), you're prompted to choose a local mirror and update the system. Both of which you should do. The local mirror only applies to Mint-specific packages. The Mint welcome wizard (which you can come back to at any time) introduces the basics and provides shortcuts to common configuration options.

Unlike Ubuntu, Debian doesn't go out of its way to make it easy to install newer versions of Nvidia's proprietary drivers. But if you have an old GPU, it might be better off with the free Nouveau driver installed by default. If you want the older version of Nvidia's driver from the Debian repos, that's easy:

```
$ sudo apt install nvidia-driver
```

But if you want a newer one, you'll have to do some work (see **https://forums.linuxmint.com/viewtopic.php?t=405012**). In general, Debian is more of a power-user distribution than Ubuntu. This has the advantage that it's usually possible to fix any problems, but the disadvantage that it's not always easy to fathom where to begin to look for a solution. Fortunately, Mint's built-in tools (such as the *Software Manager* and *Log Viewer)* can often show you the way.

If you **want** to **try** a **different** desktop, such as **MATE**, you'll **find them in** the Soft**ware Manager**.

» ENABLING PAE

To install a PAE kernel, first check that your CPU supports the required instructions by running `cat /proc/cpuinfo` in a terminal. Look for `pae` in the flags section. If it's not there (and you don't have one of those Banias CPUs we mentioned), don't try to install a PAE kernel. Otherwise, do so with:

```
$ sudo apt install linux-headers-686-pae linux-image-686-pae
```

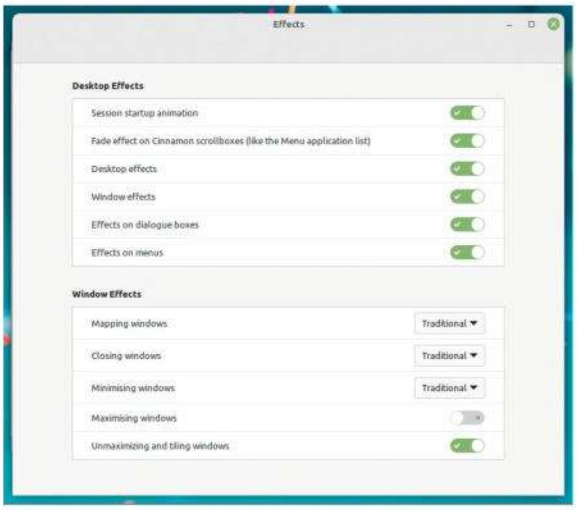
In theory, if you boot the live system in Forced PAE mode, a PAE kernel should be installed out of the box and the required boot parameters added. This might not happen, though, in which case, you need to edit **/etc/default/grub.cfg**. Add `forcepae` to the `GRUB_CMDLINE_LINUX_DEFAULT` line, then run `sudo update-grub` to generate a new configuration file. If you reboot and the system freezes, then your machine does not (correctly) support PAE. You can press `E` on the boot menu to boot without forced PAE, then undo your changes to **grub.cfg** to enable booting again.

Vintage PC options

We look at some alternative web browsers, and summarise the ultra-lightweight distro scene for particularly old systems.

Having got LMDE installed on our woefully underpowered EeePC, it became apparent that the machine wasn't going to be very useful. Except for pandering to our sense of nostalgia. According to *Systemd-analyze*, it took a minute to boot (as far as the login screen), a far cry from the sub-20s boot times we've come to expect. Initial memory usage (as gauged by the `free -h` command) was 500MB or so. Following the welcome screen's recommendations, we let it chug through a handful of updates.

The simple act of starting *Firefox* brought the machine to its knees *(kneees? – ed),* and even our beloved **linuxformat.com** took ages to render. And then began the swapping. Our install set up a 1GB swap partition (which is used as memory when actual memory runs low). This made the machine not at all fun to do anything with, and soon we saw applications being randomly closed as out-of-memory (OOM) errors proliferated. Oh, and it still had its original battery, for which a full charge (which took several hours) lasted approximately half an hour.



If LMDE's Cinnamon desktop is laggy and unresponsive, disable some of these options.

There are several lightweight web browsers that can mitigate the crawl to some extent, but don't expect miracles. Some of these are small projects, such as *Pale Moon* (**https://palemoon.org**), which don't have dedicated security teams. And for which there is some lag inheriting security fixes from Mozilla *(Pale Moon* is a *Firefox* fork that retains the traditional look but incorporates as many new features and defences as it can). *Pale Moon* was no use for our hardware, because it stopped doing 32-bit builds some time ago. And it has recently announced that its 64-bit builds will require AVX extensions (part of the -v3 microarchitecture requirements we mentioned earlier). So much for that, then.

A more promising candidate is *Midori* (**https://astian.org/midori-browser/**), which used to be the default browser on Bodhi (which we'll get to). The authors describe it as "a lightweight yet powerful web browser, which runs just as well on little embedded computers named for delicious pastries as it does on beefy machines with a core temperature exceeding that of planet Earth". Sounds ideal. On Ubuntu, this is easy to install from the Snap store, but since Mint doesn't include the Snap daemon by default, we thought we'd install it via Flatpak. The official instructions say this should be as simple as:

```
$ flatpak install flathub org.midori_browser.Midori
```

However, it turns out that the Flathub repository no longer supports 32-bit Flatpaks. So, if you're on 32-bit (or opposed to these modern packaging formats), you can find traditional DEB files available from the website. Or you can add their repository if you want a more permanent solution. We installed the DEB with an old-fashioned:

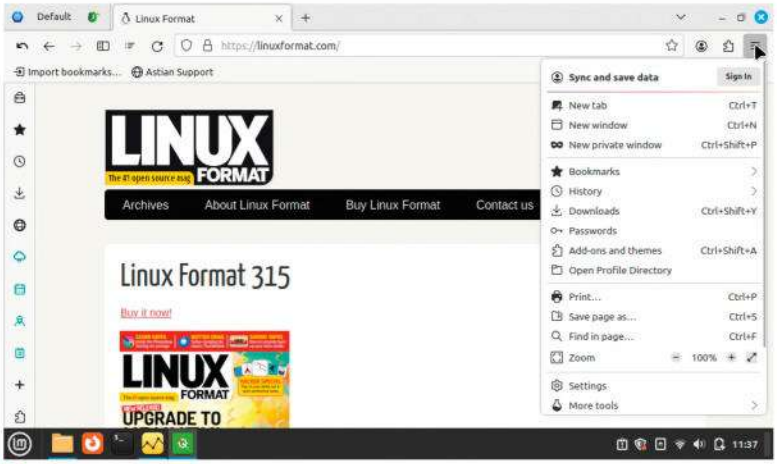
```
$ sudo dpkg -i midori_11.*_i386.deb
```

LEVERAGING UBUNTU PRO

The lightweight desktop versions of Ubuntu 18.04 (Lubuntu, Ubuntu MATE, Xubuntu and so on) were the last official Ubuntu releases to support 32-bit x86 silicon. It's possible to run the flagship release on 32-bit by upgrading 16.04, but we'd wager the Gnome desktop isn't suited to such hardware. And things might end up getting messy if you try to uninstall it. Ubuntu 18.04 LTS went EOL in May 2023, but if you enrol those machines into Ubuntu Pro (you can do three for free), this is extended until 2028.

This trick works on distros based on Ubuntu, too, including Bodhi and Linux Mint (with some effort and risk, see **https://forums.linuxmint.com/viewtopic.php?t=391484**), but is unsupported. If you're feeling brave, do your research, then install *Ubuntu-advantage-tools* and sign up at **https://ubuntu.com/pro**. Naturally, you may be opposed to this service for privacy reasons, and that is fine.

Just because an Ubuntu Pro-enabled machine will receive updates, this does not imply that it will eventually get the latest versions of all the software installed on it. The updates are almost entirely security and bugfixes. Applications in general remain at the major version as when the standard maintenance window expired, so you won't see any new features being added.



Midori is fast and fully featured, though there is some advertising on the default homepage.

Midori certainly loaded much more quickly than *Firefox,* and used a lot less memory. But it still wasn't what we would call a smooth experience.

So, it's certainly possible to run Linux on hardware that is approaching two decades old, but it's unlikely to transform those machines into anything of utility. For slightly less vintage machines, say a decade old, the outlook is much more hopeful. There's a plethora of lightweight distros and desktops that will absolutely (in conjunction with our hardware tips from earlier) breathe new life into those middle-aged machines. Just remember that old hardware fails (new hardware does, too, just less often), so if you go with LMDE, back up regularly. The *TimeShift* utility (similar to Windows System Restore points or Apple's *Time Machine)* is worth setting up, too, in case your ageing hardware proves incompatible with a future system update.

Lightweight distros

We were really impressed with LMDE 6. But it just wasn't suited to our Eee PC. Or, very likely, anything that doesn't meet its system requirements. So, let's finish by looking at some options for very old hardware. An honorary mention must first go to MuLinux (**https://micheleandreoli.org/public/Software/mulinux/**), which you can, in fact, run on a 386. As far as proper Linux goes, though, we'll never tire of recommending Bodhi Linux (**https://bodhilinux.com**). Bodhi is unique in that it has its own desktop, Moksha, which is incredibly lightweight and based on the Enlightenment window manager. This gives it a vaguely steampunk look, and offers tremendous configurability.

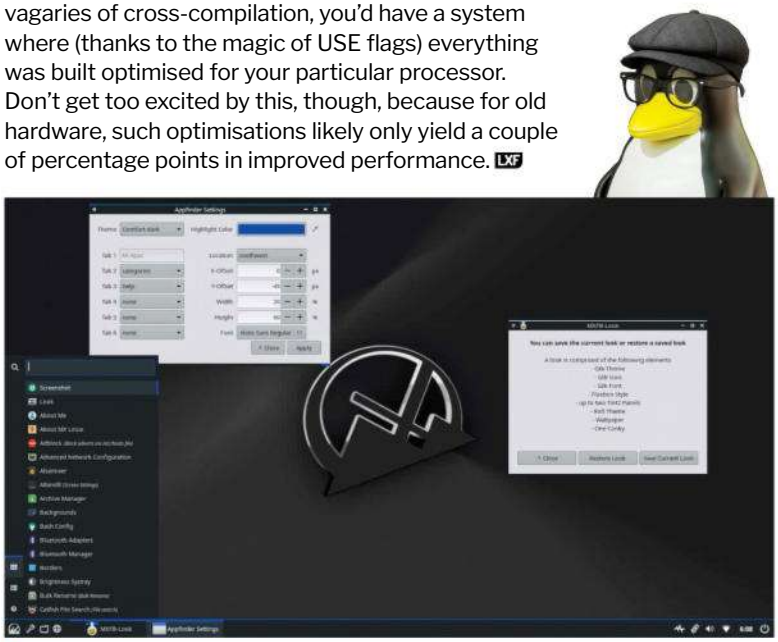
The standard release (7.0) of Bodhi is a 64-bit only affair, but of interest to us is the 32-bit Legacy release (5.1), based on Ubuntu 18.04. Like LMDE, Bodhi's Legacy release doesn't require PAE. Being based on Ubuntu 18.04, it is past the official support window, but there are ways around this *(see box, opposite).* The Bodhi-specific parts do receive occasional updates, too. We ran a feature on Bodhi Linux back in **LXF291**.

Another lightweight favourite of ours is AntiX (**https://antixlinux.com**), which is based on Debian. Thus it inherits support until 2028. AntiX uses the lightweight IceWM desktop and includes a hacker-style *Conky* display of system metrics thereon. AntiX is also one of very few *Systemd*-free distros, offering either tradition SysV-style initialisation or the modern *runit* service manager. Most people by now have grown tired of *Systemd* bashing, but if you feel it's bloated and has its tendrils where tendrils shouldn't be, AntiX is worth a look.

AntiX has a sibling distro called MX Linux (**https://mxlinux.org**), also based on Debian. MX describes itself as a midweight distribution, and offers a choice of three desktop options. The flagship offering is Xfce, then for more capable hardware there's a KDE Plasma edition. Most relevant for this feature, though, is the Fluxbox release. Fluxbox is an incredibly lightweight desktop similar to OpenBox and IceWM. MX Linux is a stylish affair with bold icons and theming.

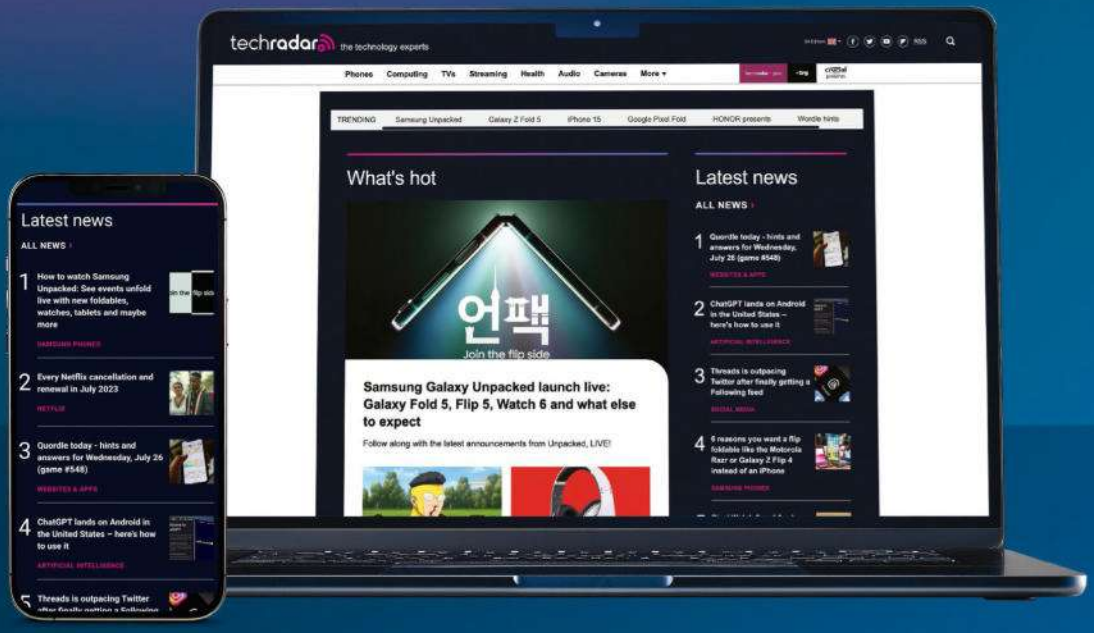
If you want a slightly more advanced OS (or are working with very old hardware where a traditional GUI wouldn't be appropriate), Arch Linux 32 (**https://archlinux32.org**) is perhaps for you. It's a community-supported fork of Reddit's favourite Linux distro, started when Arch stopped supporting 32-bit machines. There are a few subarchitectures available: i486, i686 and pentium4. Confusingly, pentium4 means any CPU supporting SSE2, which includes many Pentium 3 and Pentium-M chips. We have an even older EeePC, the 700 Surf model (with 2GB SSD and 512MB RAM), that has happily been running Arch 32 for five years. We use the Wayland-powered tiling window manager *Sway* and run everything from the terminal. We could do a whole feature on the thrills of *nnn* (a file manager), *w3m* (an image viewer) and *urxvt* (a Unicode terminal via which you run the other two).

Gentoo Linux also supports old hardware, with the slight caveat that you have to compile everything yourself. Doing this on old hardware is not going to be fun (or indeed possible). So, you'd want some sort of distcc setup, where build jobs are offloaded to a more capable machine. Once you've done battle with the vagaries of cross-compilation, you'd have a system where (thanks to the magic of USE flags) everything was built optimised for your particular processor. Don't get too excited by this, though, because for old hardware, such optimisations likely only yield a couple of percentage points in improved performance. LXF



MX Linux with Fluxbox is both stylish and lightweight.

Meet the technology experts



- The world's **most** comprehensive technology website
- An **unrivalled** mix of news, opinions, reviews and features
- **All-new design**, new homepage, new features and special reports
- Backed by **over 300 years** of editorial experience

techradar
the technology experts

www.techradar.com

$40 million Raspberry Pi IPO set for June

London Stock Exchange to be hit by Pi.

The trading subsidiary of the Raspberry Pi Foundation has confirmed a June 2024 date for its $40 million initial public offering (IPO). The confirmation comes via a 'Confirmed Intention to Float' document published on the London Stock Exchange (LSE).

The IPO is comprised partly of existing shares being sold by certain shareholders, including the existing principal shareholder Raspberry Pi Mid Co Limited, which is a wholly owned subsidiary of the Raspberry Pi Foundation, with Raspberry Pi Foundation CEO Philip Colligan as director. New shares will be issued for sale by Raspberry Pi as a means to raise $40 million, which will be used for engineering capital expenditure, to enhance the Raspberry Pi supply chain, and for general corporate purposes.

In the announcement, existing shareholders Arm and Lansdowne Partners entered into an agreement in which the two agreed to purchase shares. Arm will purchase $35 million worth of shares, and Lansdowne will purchase up to a maximum of $20 million in shares.

Paul Williamson, senior vice president and general manager, IoT Line of Business, at Arm commented: "With a shared vision to lower barriers to innovation and make computing accessible for everyone, Arm and Raspberry Pi are natural collaborators."

If you want to know how the Raspberry Pi Foundation expects this to affect its education work, you can see its full statement here: **https://bit.ly/lxf317pi**

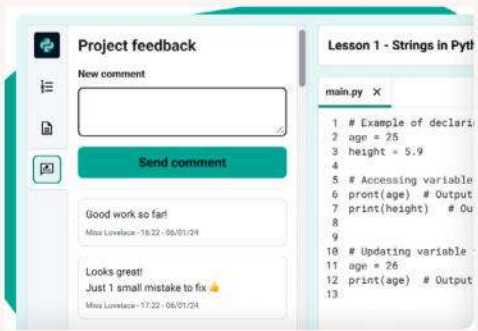
Behold Raspberry Pi Towers!

CREDIT: Otaylor, CC BY-SA 4.0 DEED, https://commons.wikimedia.org/wiki/

Easier coding

With management tools

The open source default code editor for the Pi has just got even better, with added code management tools for classrooms. This is alongside improved Python library support. The new teaching tools enable educators to share activities and leave direct pupil feedback. For more: **https://bit.ly/lxf317code**



Coding just got even more sociable.

CREDIT: Raspberry Pi Foundation

Get together!

Clubs are back, baby

The Raspberry Pi Foundation has announced that its second ever Clubs Conference will be held on Saturday 30th November and Sunday 1st December 2024 in Cambridge, UK. It will be a weekend of learning and connecting for educators and volunteers in Code Club, CoderDojo and other initiatives. Find out more: **https://bit.ly/lxf317club**

Everyone can come join in the fun (and education).

Les Pounder works with groups such as the Raspberry Pi Foundation to help boost people's maker skills.

» TAKING A HAT FOR A DRIVE

The Raspberry Pi M.2 HAT+ has finally been released and, at £12, it is rather a good deal. Providing a dedicated M.2 interface for 2230 and 2242 NVMe SSDs, this HAT+ board connects to the PCIe expansion port for power and data. If you need GPIO access, the included extension header is just tall enough. Flash the latest firmware to your Raspberry Pi 5 and set the PCIe speeds to Gen 3, and you have the ultimate speed boost for your Pi.

I took one for a spin over at Tom's Hardware and I have to say that I am impressed – and a little disappointed. The access to NVMe storage is the biggest win. It is cheap, plentiful and low cost. Going back to microSD cards now feels like driving a Ford Fiesta after driving a Formula One car. The GPIO issues can be mitigated, but the extension header is only just tall enough. The biggest issue – don't laugh! – is the plastic screws. Yes, the M2.5 standoffs are all plastic and that means they will not stand up to repeated screwing and unscrewing. Replace them with some brass versions, though, and all is good.

If you don't want to run an NVMe drive (and why not?), then you can connect other M.2 devices to the interface – just make sure that they work on the Raspberry Pi 5 before committing. If you need NVMe and an extra M.2 slot, perhaps Pimoroni's NVMe Base Duo or Pine Boards' HatDrive! Dual boards might be to your liking? These boards offer dual M.2 slots, and with the latest Raspberry Pi 5 firmware, that means we can have multiple NVMe/M.2 devices on the PCIe bus.

Ubuntu 24.04 LTS Pi

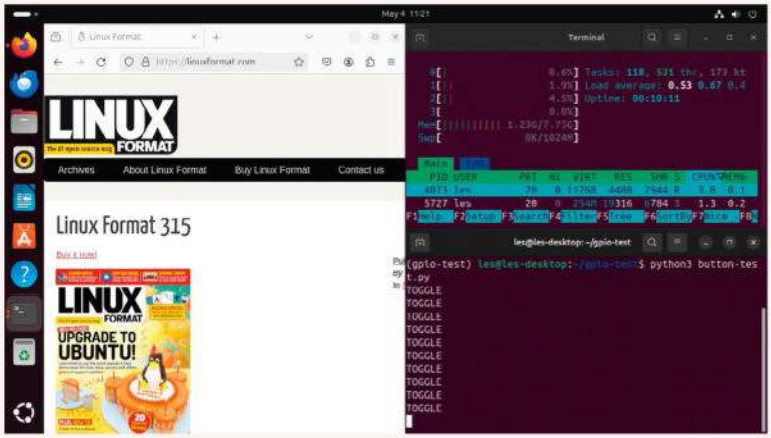
Les Pounder has been using Ubuntu since 2005, which makes him feel incredibly old. Can Ubuntu 24.04 on the Pi 5 make him young again?

IN BRIEF

The first LTS (Long Term Support) release to support the Raspberry Pi 5, Ubuntu 24.04 is a great-looking OS that can easily turn your Pi 5 into a low-power desktop computer. There are some technical glitches on the Pi 4 and 5, but if you can work around them, you get a great OS experience.

Ubuntu 24.04 is here and we've been running it for a few weeks on a work laptop – it is a slick and functional experience. So, how does the first LTS (Long Term Support) release for the Raspberry Pi 5 perform? We took it for a spin on a Raspberry Pi 5 and Pi 4 8GB. Both boards are running at stock speeds for a fair comparison.

We hit a well-documented problem. It seems that Ubuntu 24.04 has a few issues with a myriad of microSD, USB SSD and NVMe drives (in the case of the Pi 5). The tldr; of it boils down to "your mileage may vary". We've had success with microSD on the Pi 4, and USB SSD on the Pi 5. You may not. Canonical is aware of this issue, and a bug has been logged on Launchpad.



The Ubuntu desktop is always a thing of beauty, and the new quarter-tiling feature helps keep all of our windows neat and tidy.

No matter how you get it installed, Ubuntu 24.04 is a lovely-looking distro. Based on kernel 6.8.0, Ubuntu 24.04 is a promising LTS for the Pi. The new installer has some glitches on the Pi, but in general it looks and feels great (images for x86 and so on do not have any corruption). The Gnome-based desktop is slick. Really slick. The quick settings in the top-right are easy to use and the new workspace switcher in the top-left is sublime. Windows can now be snapped into the corners for quarter-tiling, giving us precious screen space. Loading apps is smooth if you have speedy storage. Our Pi 5 install was on a USB SSD and it ran beautifully. The microSD install (the only install that worked on the Pi 4) was acceptable but at times it could test your patience.

Web browser video playback is where the Raspberry Pi always drops the ball, and Ubuntu 24.04 sees it dropped again. At 1080p60, *Big Buck Bunny* ran on the Pi 5, but it was jerky. On the Pi 4, it was a slideshow. We only saw eight frames played in one minute of video.

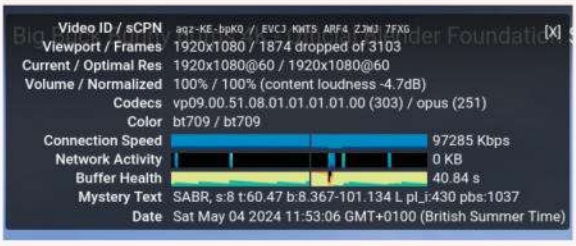
Ubuntu 24.04, like 23.10, has no support for the official Raspberry Pi cameras and we did try to install libcamera, but alas nothing happened. USB webcams are supported, so we can use Ubuntu 24.04 on the Raspberry Pi as a low-power desktop.

The GPIO is accessible, but on both the Raspberry Pi 4 and 5, the stalwart RPi.GPIO Python module is no more. You need LGPIOD and GPIO Zero if you want to make any electronic projects. We'd also follow PEP668 guidance (**https://peps.python.org/pep-0668/**) and use virtual environments to prevent damage to the OS-level Python install.

The Raspberry Pi 5 needs active cooling to keep its quad-core 2.4GHz Arm Cortex-A76 64-bit CPU cool, and Ubuntu 24.04 controls the fan in the same manner as Raspberry Pi OS.

Ubuntu 24.04 on the Pi is a mixed bag. Installation issues aside, this is a great desktop OS for those who want a low-power desktop computer. It looks great, and on the Pi 5 with a USB SSD, it runs great (we can't wait for it to be fixed for NVMe). This isn't your distro for any maker projects – stick to Raspberry Pi OS (and for those still using old HATs, stick to the older Raspberry Pi OS releases). Our hope is that the issues that we, and others, have faced are rectified in a 24.04.1 release. A respin is unlikely at this time, so updates are the way to go.

A great look and feel, sadly let down by technical issues. Give it a try on whatever microSD/USB SSD/NVMe drive it works on for you. LXF



Even the mighty Raspberry Pi 5 didn't fare well with 1080p60 streaming video, dropping 1,874 of 3,103 frames in our tests.

VERDICT

DEVELOPER: Canonical
WEB: https://ubuntu.com
LICENCE: Mixed

FEATURES	7/10	EASE OF USE	8/10
PERFORMANCE	7/10	DOCUMENTATION	10/10

Looks great and performs well on the Raspberry Pi 5. Sadly, it is let down by technical glitches.

» Rating 7/10

QIDI Tech Q1 Pro

Getting hot under the collar, **Denise Bertacchi** likes this heated model.

SPECS

Vol: 245x245 x240mm
Type: PLA, PETG, TPU, ABS (up to 350°C)
Extruder: Direct drive
Nozzle: 0.4mm (dual-metal high flow)
Plate: Coated steel flex plate, heated 120°C
Levelling: Automatic, Z offset, dual Z control
Sensors: Runout, internal temp, tangle, 1080p camera
Comms: LAN, Wi-Fi, USB flash drive
Control: 4.3-inch colour touchscreen
Size: 477x467x 489mm, 17kg

It feels like QIDI Tech has been sitting in the back of the room, taking notes while we discuss other 3D printers' flaws. The Q1 Pro fixes many of the little problems: it eliminates levelling guesswork by setting its own Z height; the spool holder is side-mounted, where it's actually reachable; and it connects easily to your house Wi-Fi without having to play Mother May with a foreign server. It also uses stock *Klipper* firmware – as too many companies slap a buggy facade on open source firmware. But it has one controversial feature: a wall-powered heater for the build chamber.

The Q1 Pro uses an active heater in the back of the enclosure to heat air as high as 60°C, which is useful for printing ABS, ASA, nylon and other advanced filaments. It is controlled from the slicer and the printer interface.

The heater is not marked as hot or hazardous, and if you poke around the interior with a hex key while it is switched on, you could receive an electrical shock. There is a safety grid over the heater, and you're unlikely to jam a finger or the corner of the build plate past the grate.

QIDI has been made aware of the issue and stated that while its printer is FCC and CE certified, it will implement a more touch-proof enclosure for future units and make a printable part available to current owners.

The QIDI Q1 Pro comes with everything you need to get your printer set up. You get tools to maintain the printer, a scraper and a USB stick. There's also a small sample coil of black PLA to start your printing journey.

There's a removable purge collector in the back, with a brush for wiping the nozzle, which seems odd on a single-colour printer. It might be an indicator that QIDI is working on a colour upgrade, or it could just be a way to keep the nozzle pristine for more accurate bed probing.

The lid is removable to provide air flow for PLA, PETG and TPU. The chamber thermometer averaged 35°C while printing with lid off and door closed. Opening the door dropped the temperature another 5°. The door opens 180° and has hinges that stay put, which keeps the door out of your way should you need to run it open.

A high-flow, wear-resistant nozzle with a copper-plated throat and hardened steel tip is installed on the printer. It's nice to see a 3D printer come with a durable, pro-grade nozzle instead of cheap brass. Replacements are affordable, and start at a couple of pounds. They're also much easier to swap out than Bambu-style nozzles.

This thing is hot stuff – perhaps too hot, if you don't realise!

Another perk is a built-in 1080p camera for monitoring prints or even doing timelapses. The camera feed pops up immediately when you access the *Klipper* interface. The chamber is brightly lit with LED lights.

Loading filament is straightforward. Just place the filament into the tube near the spool holder and push it through until it bumps against the nozzle. Press the 'three lines' button on the bottom of the screen to get to the loading menu, and then press Load. It warms up the printer and advance the filament.

We ran a Benchy using Speed Benchy rules: 0.25mm layer height, two walls, three top and bottom layers, 10% infill. We turned off combing and Z hop, and let her rip using the default speed of 300mm/s. The Benchy is a bit stringy, but the layers are fairly even, the curves nice and there's no ringing. This Benchy printed in an impressive 20 minutes and 18 seconds, which isn't enough to make our top 10 fastest 3D printers, but is faster than most. LXF

Not the fastest printer tested but the results are very sound.

VERDICT

DEVELOPER: QIDI Tech
WEB: https://uk.qidi3d.com
PRICE: £399

FEATURES	8/10	EASE OF USE	9/10
PERFORMANCE	7/10	VALUE	8/10

Fixes small problems seen on new core XY printers, but has a questionable heated chamber that some feel is dangerous.

» Rating 8/10

PYTHON

Build your own fortune teller

Mystic **Les Pounder** predicts that you will learn lots of Python and Pygame knowledge in this tutorial.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his adventures and projects at **http://bigl.es**.

YOU NEED

- **A Pi 3,4 or 5**
- **All of the code examples, images and audio can be found in the GitHub download package at https://github.com/lesp/LXF317---Fortune-Teller/archive/refs/heads/main.zip**

One of my earliest commissioned projects was for a coin-operated fortune teller, which lives in an arcade in Blackpool. I wanted to make a modern version, so I dug out a Raspberry Pi 4 and dusted off my old *Pygame* knowledge. We are making a fortune teller video game in 55 lines of Python.

Prepare thy Pi

First, we need to set up our environment. Raspberry Pi OS Bookworm follows PEP668, which forces us to set up virtual environments (virtualenv) to install Python packages. This is so our manually installed packages do not conflict with the Python setup for the OS.

Open a terminal and first update the repositories and software on your Raspberry Pi:

```
$ sudo apt update && sudo apt upgrade -y
```

Create a virtualenv called **fortune-teller**. It may take a while as Python sets things up behind the scenes:

```
$ python -m venv fortune-teller
```

Change directory to **fortune-teller** and then activate the virtual environment. This changes the prompt to show that it is active and that we are now working with the version of Python installed within.

```
$ cd fortune-teller
$ source bin/activate
```

The final step in the Python setup is to install *Pygame,* a game and multimedia creation tool:

```
$ pip install pygame
```

Make two directories in the **fortune-teller** directory: **audio** and **images**. Copy the respective images and MP3s to the relevant directory.

Open *Thonny,* found under Programming in the main menu, and create a new blank document. Save it as **fortune_teller.py** to the **fortune-teller** directory.

Our first section of Python has four imports. **Pygame** is used to make an interactive game from the code, **OS** enables us to work with files in the operating system. **Random** randomly selects an audio file, while **sleep**, from the time module, is used to pause the code.

```
import pygame
import os
import random
from time import sleep
```

We need to initialise *Pygame* and its audio mixer:

```
pygame.init()
pygame.mixer.init()
```

Pygame requires us to set the size of the screen; this doesn't have to match our monitor. We've chosen 1,024x1,024 to match the fortune teller image, storing the values in corresponding objects for width and height. Then we set the screen to use those details:

```
SCREEN_WIDTH, SCREEN_HEIGHT = 1024, 1024
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
```

We now move on to preloading the images ready for use in the project. First we need to tell the code where

» CREATING AUDIO AND IMAGES

This project uses a single image of five fortune tellers. How you make this image is up to you. We selectively removed the colour (saturation) from four fortune tellers, and over-saturated the remaining. All this was done using *Gimp.* We saved six versions of the image. One greyscale, and five with one fortune teller highlighted.

The audio files were made with *Audacity* and our 'beautiful' BBC presenter voice. We used filters (amplify, delay and change pitch) to give the audio a mystical tone. All audio files were saved as MP3.

Pygame is pretty versatile when it comes to media. Given that it is a game creation and multimedia tool for Python, it has to be. We first discovered *Pygame* via PyCon UK in 2014, via a hands-on session. We loved how it worked so 'Pythonically', enabling us to drop it into a project with relative ease. Since that day, we have made demoscene-type intros and even a camera controller using a webcam. The documentation is excellent, but don't rush head-first into it. Take your time to learn the basics before moving on. We made that mistake and it costs us a bunch of time.

Read more about *Pygame* at **www.pygame.org/news**.

the images are. For this we create an object, **image_folder**, then create a list of filenames that have the folder name joined to them. This creates a relative path to the images:

```
image_folder = "images"
image_files = [
    os.path.join(image_folder, "0.jpg"),
    os.path.join(image_folder, "1.jpg"),
    os.path.join(image_folder, "2.jpg"),
    os.path.join(image_folder, "3.jpg"),
    os.path.join(image_folder, "4.jpg"),
    os.path.join(image_folder, "5.jpg"),
]
```

Using a **for** loop, we load the images ready for *Pygame*.

```
images = [pygame.image.load(file) for file in image_files]
```

Create an object, **image_index**, to handle which image is currently selected; we'll update this object as the code runs. Then create **image_timer**, which will count how long an image is displayed (in milliseconds/ticks).

```
image_index = 0
image_timer = pygame.time.get_ticks()
```

To set up the game loop, we need to create a clock, and set the game running and the **image_sequence** running using boolean operators. Essentially, this says, start the clock and run the game and images:

```
clock = pygame.time.Clock()
running = True
image_sequence_running = True
```

Just like we did with the images, we now need to load up the audio files ready for playback:

```
audio_folder = "audio"
audio_files = [
    os.path.join(audio_folder, "0.mp3"),
    os.path.join(audio_folder, "1.mp3"),
    os.path.join(audio_folder, "2.mp3"),
    os.path.join(audio_folder, "3.mp3"),
    os.path.join(audio_folder, "4.mp3"),
]
```

On to the main part of the code. This is what runs our fortune teller game. The **while running** loop has just been set to **True**, so the game starts. It looks for an event, in this case the user closing the window. If that happens, the **running** object is updated to **False** and the game ends.

```
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

The next part of the conditional test looks for a keypress (**keydown**) and if that is **SPACE** (Spacebar), it stops the image animation on the currently chosen image and sets **image_sequence_running** to **False**.

```
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                image_sequence_running = False
```

The next section handles the image animation. Essentially, it swaps the image to the next in the sequence every one second. This gives the illusion of a moving image. It uses a timer of 1,000 ticks (1,000 milliseconds, one second) before increasing the value stored in the **image_index** to cycle through the images.

```
    if image_sequence_running:
        if pygame.time.get_ticks() - image_timer > 1000:
            image_index = (image_index + 1) % len(images)
            image_timer = pygame.time.get_ticks()
```

To show the image, we need to fill the screen with black (**0,0,0**) and then using the current image, we need to blit the image to the screen. Blitting is basically writing the image to the video buffer. We then flip the display to update:

```
        screen.fill((0, 0, 0))
        current_image = images[image_index]
        screen.blit(current_image, ((SCREEN_WIDTH -
current_image.get_width()) // 2, (SCREEN_HEIGHT -
current_image.get_height()) // 2))
    pygame.display.flip()
```

When the user presses Space, the image running sequence, the animation, ends and an audio file is randomly chosen and loaded ready for playback.

```
    if not image_sequence_running:
        random_audio_file = random.choice(audio_files)
        pygame.mixer.music.load(random_audio_file)
```

To play the audio file, we simply call the **play** function. Whatever is loaded into the mixer will be played. We added a seven-second pause to enable the file to be played without the animation restarting. Then we stop all playback and use **True** to restart the image sequence.

```
        pygame.mixer.music.play()
        sleep(7)
        pygame.mixer.music.stop()
        image_sequence_running = True
```

Finally we come out of all the loops and conditional tests to quit *Pygame*. Typically we will never hit this line of code, but if everything crashes, we have a means to exit the code:

```
pygame.quit()
```

Save the file as **fortune_teller.py** to the **fortune-teller** directory. Go back to the terminal and run the code via the virtualenv:

```
$ python fortune_teller.py
```

The animation plays, awaiting your fortune teller selection. Press Space and your fortune is told. LXF

Our fortune teller runs on pure Python and it can predict only nice things!

PI CONNECT

Credit: https://connect.raspberrypi.com

All of your Pis are under your control

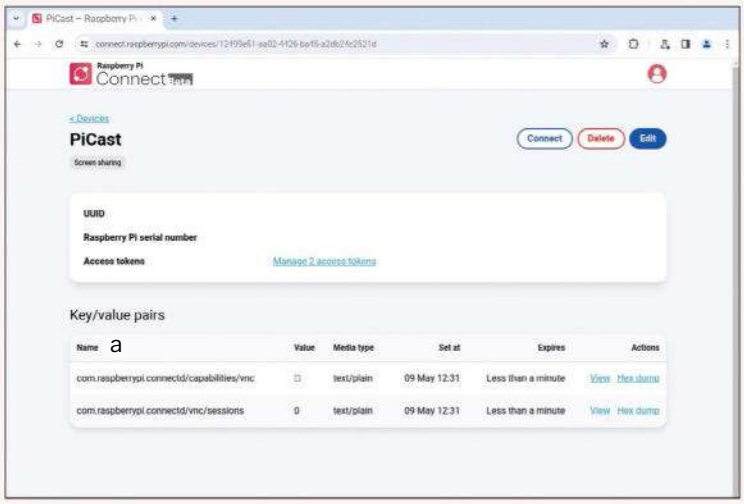
No one would say **Les Pounder** has gone power mad, but he's cackling like a Bond villain now he can control all the Pis.

OUR EXPERT

Les Pounder is a Pi project profesorado and is connected up to 3.14 Raspberry Pis at any one time. He loves maths dad jokes, too.

Raspberry Pi has announced *Raspberry Pi Connect*, its take on a remote desktop session, like VNC and RDP. The difference with *Connect* is that it runs from a web browser, so we can be anywhere in the world and log in to our Raspberry Pi back home.

This project only works on Raspberry Pi 4, Pi 5 and Pi 400, as it requires a 64-bit version of Raspberry Pi OS. Other than that, all you need is network connection for your Pi and this guide to control your Raspberry Pi.



The public front-end to the new service where you can connect to your Pis.

Get up to date

Power up your Raspberry Pi to the desktop and open a terminal. Update and then upgrade the software on your Raspberry Pi. The *update* command checks your list of installable packages against the latest list on the remote server. The *upgrade* command then downloads and installs the updates. Using the **-y** switch automatically accepts the installation all the updates.

```
$ sudo apt update && sudo apt upgrade -y
```

Pi connect

From the terminal, install *Raspberry Pi Connect*:

```
$ sudo apt install rpi-connect
```

Reboot your Raspberry Pi for the installation to take effect. *Raspberry Pi Connect* is a background service that waits for connections. The service starts when the Raspberry Pi boots.

In the top-right corner of the desktop, look for the *Raspberry Pi Connect* icon. Click on the icon and select Sign In. Follow the verification URL and sign in, or sign up to the Raspberry Pi ID service.

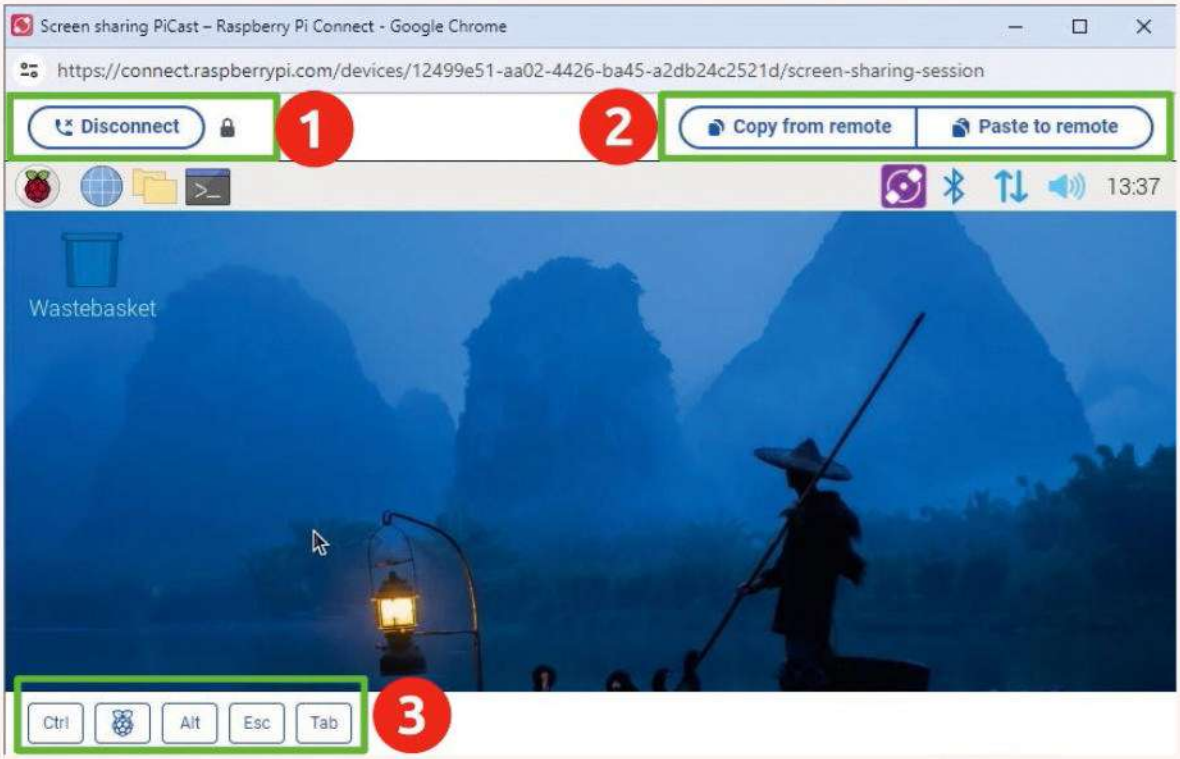
Give your Raspberry Pi a device name. Make this descriptive; for example, if it is a weather station, name the device Weather-Station. The Raspberry Pi is now available for connection. On another computer, open a web browser to Raspberry Pi Connect and sign in.

Select the Raspberry Pi from the list of devices. The browser connects to your running Raspberry Pi in a few seconds. You can repeat the above steps to add more Raspberry Pi devices to the list, enabling control of many Pis from anywhere in the world.

The *Raspberry Pi Connect* interface wraps around the Raspberry Pi OS desktop in much the same way that *RealVNC* offers extra functions for your session. *Raspberry Pi Connect* keeps it simple with only three areas for use *(see screenshot, opposite page):*

1. **Disconnect** This ends your session but the Raspberry Pi remains powered up. Hovering over the padlock shows the connection status. For our local connection, it was peer to peer.
2. **Copy and Paste** These functions enable copy and paste functionality between the remote Raspberry Pi and client machine. They copy/paste from the clipboard of the respective machine.
3. **Keys** These keys mimic pressing the respective keys on your keyboard. The Ctrl, Super (Raspberry) and Alt keys are sticky, while Esc and Tab release when pressed.

The browser window scales to match the resolution of your client system. This can be beneficial, but we have encountered a bug where the top section of *Thonny's* dialog box was missing. This was a big issue because that is where we need to write the filename when saving. We got around this by simply typing the name and pressing Enter.



Raspberry Pi Connect in service, controlling our remote Pi.

Other than that issue, everything works as if we are sitting in front of a Raspberry Pi. On our home network there was a little lag, noticeable when moving windows and clicking on UI elements. That said, there was nothing horrifically slow. A delay is introduced should you use Raspberry Pi's TURN server.

The Pi Foundation explained that, "At the moment, the Raspberry Pi Connect service has just a single relay (TURN) server, located in the UK. This means that if *rpi-connect* chooses to relay traffic, the latency can be quite high." You're able to check what routing is going on with your Pi Connect by hovering the mouse cursor over the padlock icon in top-left of your browser URL address. This says if the connection is routed or not and enables you to know if you can improve things by changing your network setup.

Currently this service is free and the Pi Foundation does outline that its intention is for it to remain free for individual users with non-relayed connections and no limit on the number of devices. This leaves the door open for commercial charges and it's going to monitor the situation with running its own TURN server, as that's where the pinch point on bandwidth will fall.

If you need more details, you can check out the detailed documentation (**www.raspberrypi.com/documentation/services/connect.html**) or you seek out more help from fellow *Connect* users at the official forums: **https://forums.raspberrypi.com/viewforum.php?f=167**.

We tested using the GPIO via *Thonny* and everything went smoothly. We can see this approach being used in the classroom where space and equipment is at a premium.

Using the official Raspberry Pi Camera was also possible, something that we demonstrated on our show, *The Pi Cast*. There was a little lag, but the video stream was fine for most use cases.

We tested using a Pi 5 connected to our home network via Ethernet, a Windows 10 PC connected via Ethernet, too, and an Android smartphone connected to the same session over a 5G mobile connection.

The smartphone interface was cramped but we could use it in a pinch. Clicking the main menu was fast enough, but moving a window around the screen was laggy. Useful for remotely controlling a project, editing some code and managing your builds.

Raspberry Pi Connect shows great promise and we can see it being used in a plethora of Pi projects. LXF

» UNDER THE HOOD

If you're wondering where your little packets of data are being sent, the Raspberry Pi Foundation has a full explanation of what's going on at the blog post **www.raspberrypi.com/news/raspberry-pi-connect/**. Alongside the basics on how to get the software and set up an account, it goes into a little technical detail on the protocols it used and the ramifications of that.

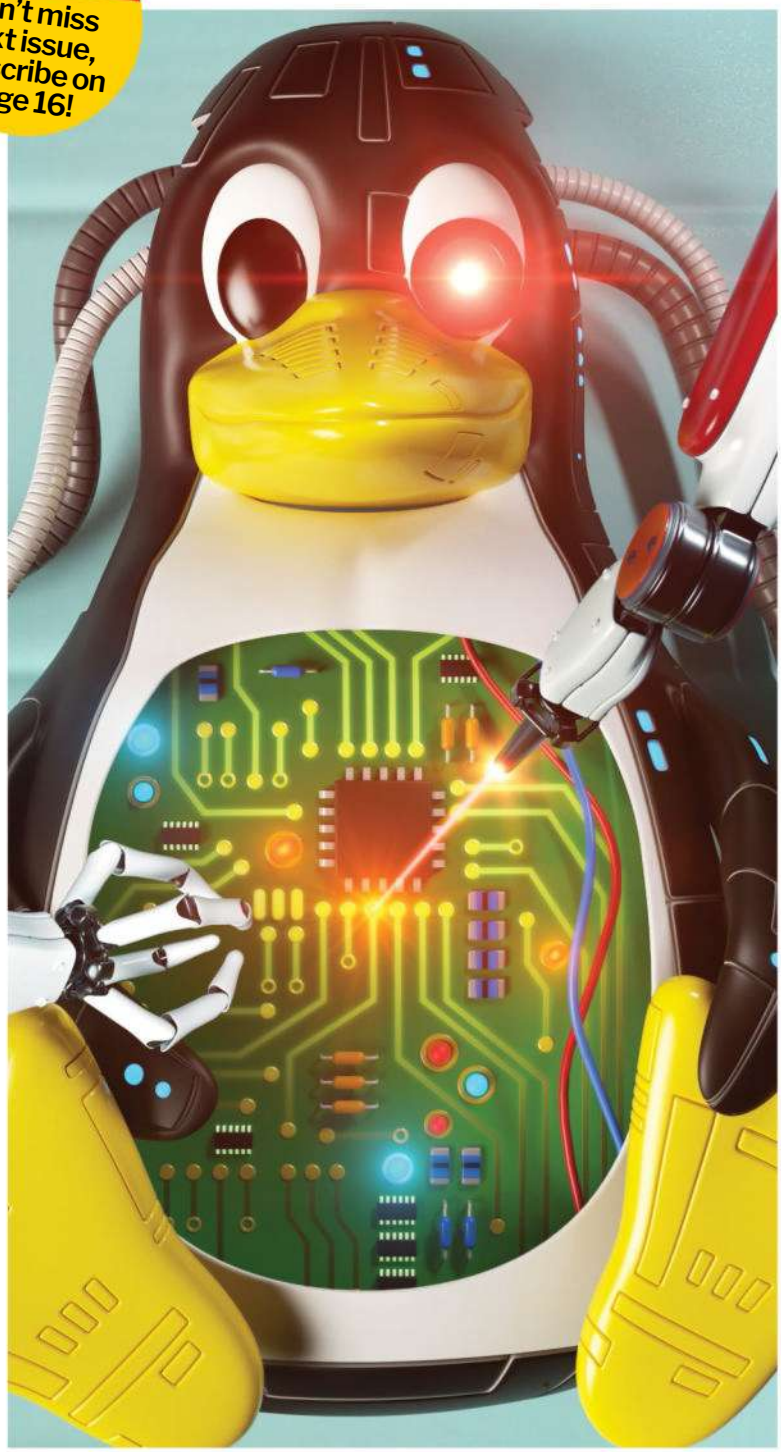
As it's browser-based, it's using the open WebRTC (see **LXF169**, **LXF213**) protocol to establish a secure peer-to-peer connection between your browser and the remote Raspberry Pi – it's the same technology as used by big names like Zoom, Slack and Google Meet.

Web development expert Paul Mucur at the Pi Foundation explains: "Our *rpi-connect* daemon for Raspberry Pi OS is responsible for listening out for new screen-sharing sessions from the Raspberry Pi Connect website, and negotiating the best possible (ie lowest latency) connection between the in-browser VNC client and a VNC server running on your device. In general, once a connection is established, no traffic need pass through our servers."

Part Two! Don't miss next issue, subscribe on page 16!

Virtual this, that & the other

Matt Holder delves deeper into the kernel to discover virtualisation, containerisation and virtual filesystems.



CREDIT: Magictorch

In this part of our *Inside Linux* series, we are looking at more of the features our kernel offers and introducing virtual filesystems and hardware detection. Last month, we covered the basics of the Linux kernel, what it does and how it makes our lives a lot easier by providing interfaces between userspace applications and the hardware itself. As we discovered, the kernel is made of multiple subsystems, each of which is split into a number of layers. This time, we are looking at the virtual filesystem layer of the storage subsystem.

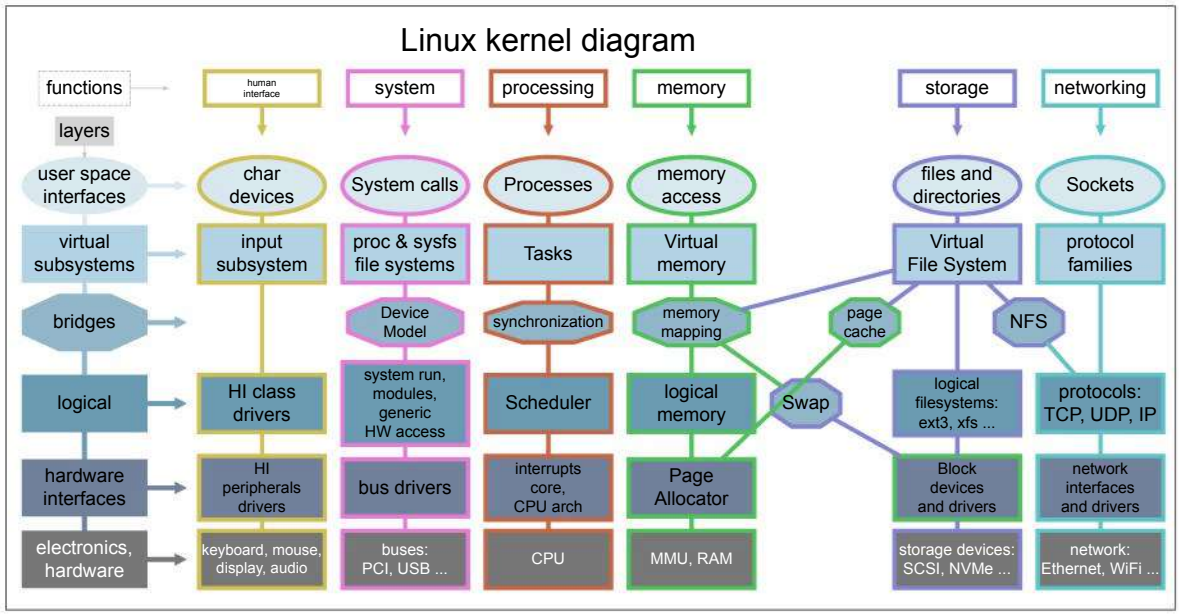
Your Linux distribution follows the concept that 'everything is a file'. This means that pictures, documents, movies, directories, writing to hardware and reading from hardware are all abstracted into filesystem operations. For example, sending data to or reading data from a serial port is done by referencing a file in a certain part of the filesystem.

Virtual first

First arriving in 2007, Kernel-based Virtual Machine (KVM) allows the kernel to function as a virtualisation hypervisor. KVM emulates a virtual CPU and memory, and has support for VirtIO. Tools such as *Qemu* are then used to provide the emulation of some of the other hardware devices as well as using VirtIO, which is a standard for the communication between the virtual machine and hypervisor to allow for high-performance networking and disk operations. Drivers are available for Windows guests as well as Linux support being widespread. We will now turn our Ubuntu installation into a hypervisor and use *Cockpit* to control it.

To set up your Ubuntu installation as a hypervisor, open a terminal and enter the following commands:

```
$ sudo apt install bridge-utils cpu-checker libvirt-clients libvirt-daemon qemu qemu-kvm
$ sudo apt install cockpit cockpit-machines
```



The kernel is made from multiple subsystems and each one is made up of multiple layers.

The first command installs the necessary tools to run virtual machines and the second installs *Cockpit,* which can be used to set up and use virtual machines. Once installed, open your web browser, navigate to **https://127.0.0.1:9090** and use your username and password to log in to *Cockpit*. Spend some time learning about the tasks you can accomplish, such as adding users and performing upgrades. Select the Virtual Machines menu on the left-hand side and use the displayed option to create a new VM.

Once the options have been filled in, hit the Create button and the VM is provisioned in the background. You can then select it from the *Cockpit* VMs page, and access any settings and view its virtual console from the web browser. It is also possible to install the **virt-viewer** package to use a separate program rather than the web browser. We have only scratched the surface of KVM here and it is capable of so much more, including extra storage pools, different networking configurations, shared directories between host and guest, snapshots and pass-through to allow hardware devices and guests to interact natively with each other.

Get in the CGroups

Also back in 2007, control groups (CGroups) were added to the Linux kernel. This technology is used to provide control over groups of processes, where each group can contain one or more processes. Resource limits can be placed on each CGroup and they can be prioritised easily, similarly to using the *nice* command to change the priority of a single process. CGroups can also be used to provide accounting of resource usage and they can be checkpointed, so that a group of executables can be stopped and restarted while retaining their previous state.

Combining the CGroups functionality with namespaces allows for processes to be segregated from each other. This was really important in the development of containerisation, which is used everywhere these days. *Docker, Podman* and *Kubernetes* all utilise CGroups and namespaces. When containers are started, they utilise their own CGroup so that resources are completely separated from each other. Clever techniques, such as using virtual Ethernet devices, allow for the networking to be connected between different CGroups.

Firewalls

The kernel also has built-in firewall functionality. This can be used for tasks such as creating a router for your home or office or protecting your laptop from other users using the same open network as you.

There have been a number of filtering systems developed for the kernel. The Berkeley Packet Filter (BPF) system uses rules compiled to machine code, which is executed on packets. It is now possible to compile C and other languages to compatible binaries.

The *iptables* userspace tool can be used to configure firewall rules using a specialist-looking syntax that certainly takes some getting used to. It is incredibly easy to configure your firewall incorrectly and cause issues with your own device. Superseding *iptables* is *nftables*, which utilises a different syntax.

Both graphical and console applications exist to provide a far more user-friendly method to configure

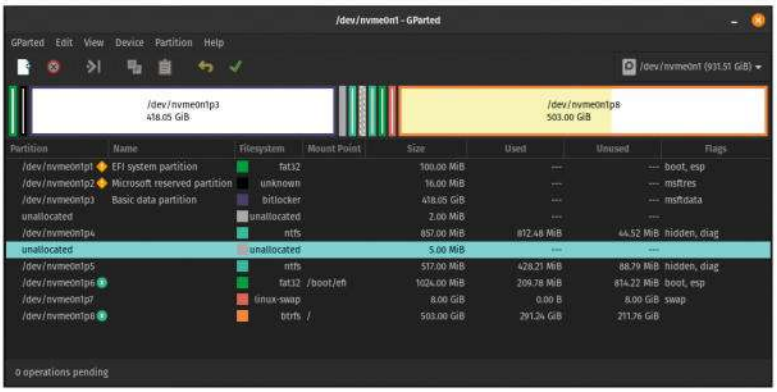
QUICK TIP

You can learn more about KVM here: www.redhat.com/en/topics/virtualization/what-is-KVM

» VIRTUALISATION DISK IMAGES

As is often the case in the Linux world, there are multiple choices available for accomplishing the same task. The KVM/*Qemu* virtualisation system supports multiple disk image types, which all have their own benefits. The first option is the RAW image. This is a file that supports the raw data being written to the virtual disk and whatever size of disk is created is the size that is used on the host disk. Performance is very good with a RAW image file, because extra processing doesn't need to be carried out. Another option is LVM. The Logical Volume Manager (LVM) is a layer that sits on top of the disk communications and allows for volumes to be created that can be changed in size and can sit on top of one or more physical disks. LVM snapshots can also be taken. The qcow2 format, which is the one used when creating images with *Cockpit,* supports copy on write, encryption and snapshots, and uses sparse file functionality to allow the file to grow in size when data is added. This means that a nearly empty disk of 300GB only utilises the data required by the guest OS. Using some of these feature may have a slight performance impact on your system compared with a RAW image.

CREDIT: https://github.com/makelinux/



Disk partitions, as shown in the GParted partitioning tool.

your firewall. Ubuntu-based distributions have the *Uncomplicated FireWall (ufw)* front-end installed. By default this is switched off, but it is simple to configure sensible defaults. Open your terminal and run the relevant commands from the examples below:

```
$ sudo ufw enable
$ sudo ufw disable
$ sudo ufw allow in ssh
$ sudo ufw allow in 8768/udp
$ sudo ufw status
$ sudo ufw delete RULE_NUMBER
$ sudo ufw default deny outgoing
$ sudo ufw default deny incoming
```

QUICK TIP

You can learn more about CGroups here: https://red.ht/4aKII2B

The first two commands in the listing above simply enable or disable the firewall. On Ubuntu-based distros, the default is that your firewall is disabled. Once enabled, you need to allow any connections to be made to the machine from outside. The third command uses a profile name (SSH in this example) to allow this service to be connected to. Command number four demonstrates how to allow specific ports and protocols to connect to your device. If you wanted to allow UDP communications on port 8768, command four is the one for you!

Moving on now to the management of existing rules, we can see that by running command five, all existing rules are shown. Adding `verbose` to the end of the command shows the default rules as well. Command number six can be used to delete rules. Before running this command, view the status to determine the rule number. Remember, if deleting multiple rules in a row, you need to view the status each time because the rule numbers change whenever a rule is deleted.

Rule numbers seven and eight are used to set the default behaviour. For example, if you wanted to completely air-gap your device and stop any internet access at all, run command seven. Command eight blocks any connections coming into the device the firewall is running on.

When using *ufw,* the biggest thing we noticed was that it wasn't possible to change the order of rules. This can be important in some instances, as one rule running before another could render the second rule useless. Firewall configuration is critical, so great care should be taken, no matter which tool you use.

Setting up your device as a simple router can be completed by running a few commands. First of all, we need to turn on IP forwarding in the kernel and then set some firewall rules. This configuration can be seen at the following link: **https://bit.ly/LXF317router**.

The same disk objects, as shown in the /dev directory.

```
matt@pop-os /dev [124]> ls -alh nvme* ttyS*
crw------- 1 root root    238,  0 Apr 27 19:20 nvme0
brw-rw---- 1 root disk    259,  0 Apr 27 20:20 nvme0n1
brw-rw---- 1 root disk    259,  1 Apr 27 20:20 nvme0n1p1
brw-rw---- 1 root disk    259,  2 Apr 27 20:20 nvme0n1p2
brw-rw---- 1 root disk    259,  3 Apr 27 20:20 nvme0n1p3
brw-rw---- 1 root disk    259,  4 Apr 27 20:20 nvme0n1p4
brw-rw---- 1 root disk    259,  5 Apr 27 20:20 nvme0n1p5
brw-rw---- 1 root disk    259,  6 Apr 27 20:20 nvme0n1p6
brw-rw---- 1 root disk    259,  7 Apr 27 20:20 nvme0n1p7
brw-rw---- 1 root disk    259,  8 Apr 27 20:20 nvme0n1p8
crw-rw---- 1 root dialout   4, 64 Apr 27 19:20 ttyS0
crw-rw---- 1 root dialout   4, 65 Apr 27 19:20 ttyS1
```

I want real files!

A virtual filesystem provides a simple way of interacting with abstract items that are linked to computers. Without this functionality, interacting with our computer's hardware would be very different.

We are going to investigate three directories on our distro's filesystem, which are provided by the kernel. All three are at the root level of the filesystem.

Firstly we will talk about the **/dev** directory. This is the location where we can interact with the hardware connected to our computers. Let's have a look at some of the entries and discuss what they are.

Any files in this directory with a name starting with **hda** represent hard drives connected via IDE. Files starting with **sda** represent drives connected via SATA or SCSI. Each system can have multiple hard drives, named **sda**, **sdb**, **sdc** and so on. Each disk can be made up of multiple partitions, which would add a number to the end of the filename. For example, **sda0**, **sda1**, **sda2**, which are also shown as files in the **/dev** directory.

Any NVMe disks connected to the system are represented by files starting with **nvme**, such as **nvme0**, **nvme0n1**, **nvme0n1p1**. These files represent the device itself and the first partition. We can see an example of disk partitioning by opening the *Gparted* tool or *Fdisk*. Be incredibly careful with these tools as you could easily delete the entire installation; these skills are best learned from a virtual machine.

Another example that we can use is that of serial ports. A serial port is represented in the **dev** directory as **ttySX** or **ttyUSBX** where the X represents a number, starting from zero. In this day and age, USB serial adaptors are more likely to be used than the hardware ports built into the motherboard. While serial ports are not widespread, some devices, such as microcontroller boards, which can be used with home automation, can be provisioned via serial ports.

Looking at the example screenshot *(left),* we can see both the files related to the NVMe hard drive and two serial devices. What can be seen here is the ownership of the files and default permissions. Taking the **ttyS0** serial device, this is owned by the root user and members of the dialout group have access as well. The first column in the screenshot shows a list of permissions. This is split into three sections, the first representing permissions granted to the owner, the second representing permissions granted to members of the group, and the third representing users who are not the owner or in the defined group. In this example, the root user has read and write permissions, the group assigned has read and write permissions, and

other users do not have any permissions at all. This means that a non-root user needs to either interact with the file using the *sudo* command, or be added to the dialout group. The dash after `rw` represents the execute flag and this shows that execute has not been assigned to either the owner or the group. The `c` at the beginning of the permission sequence shows the type of file. In this instance it is a 'character file' file.

The **/sys** directory also contains information about your computer's hardware, including the Ethernet adaptors, the hardware that controls the backlight of your display, and many other items. By writing to specific files in this directory, it is possible to change things, such as the backlight level. This is not recommended, though, because it would be very easy to damage your computer. Other pieces of code have been written to provide a user-friendly way to interact with these items.

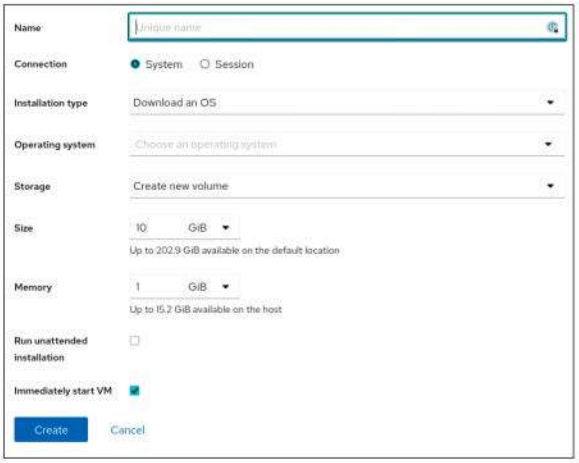
Finally, we'll have a look at the **/proc** directory. This contains information about the running system and is generated on boot or by the kernel when required.

In the next screenshot *(above-right)* you can see lots of folders and files. The information stored within the **/proc/sys** directory contains settings that can be changed in the running kernel *(see last month's article for more information).* To change the settings, *sysctl* is used. Directories within **/proc** with a numerical name relate to running processes. These directories contain a lot of information, including the command used to invoke the process, memory being used, CGroups the process is linked to, and much, much more.

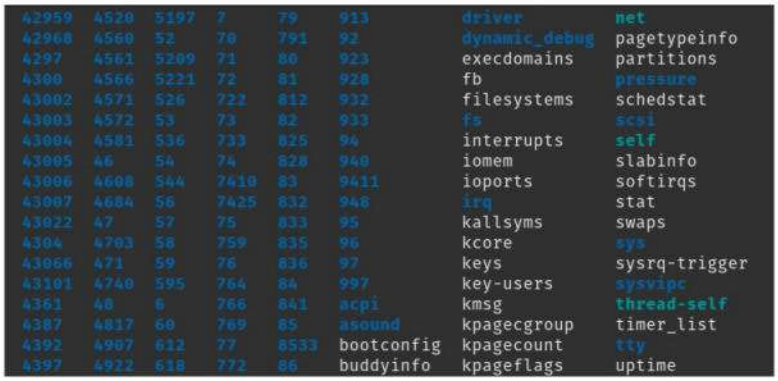
Userspace devices

Before we move on from virtual filesystems, which allow us to interact with our hardware, let's take a look at how our distros initialise our hardware. A system called *udev* is utilised to monitor hardware changes on the system and then configuration files are used to determine what should happen when the status of devices changes. A number of years back, *udev* began being developed under the banner of the *Systemd* init system. By writing custom *udev* rule files, we can accomplish useful tasks, such as starting a backup process when a specific USB hard drive is connected.

The first step to accomplishing this is to find out what the device path is to the newly connected device. We can do this by opening a terminal and running



Fill in the options and select the Create button.



The /proc directory contains lots of information about your running system.

`dmesg -w`. Now connect the memory stick or hard drive. From the output, you might see that the device is registered with the system as a file called **sda**, for example, in the **/dev** directory. You could now use `udevadm info /dev/sda` to find out more information about the device

The information from this output will be used to devise our new rule and script. Back in the terminal, create a new text file in **/usr/lib/udev/rules.d/** called **99-usbBackup.rules** and enter the following text (adjusting for the output from your USB device):

```
ENV{SUBSYSTEM}=="block", ENV{ID_SERIAL_
SHORT}=="ADD_SERIAL_NUMBER",
ACTION=="add", RUN+="/PATH/TO/SCRIPT.SH"
```

The referenced script is then run when the specific device is plugged in. When writing the script, don't forget to make it executable and end any long-running process with an `&` so it runs in the background.

We hope you have enjoyed this second dive into the kernel and it has given you an understanding of what an amazing piece of software development it is. Utilising the built-in firewall functionality and a user-friendly front-end is an excellent way of keeping your distro safe from any bad actors – especially if using an open Wi-Fi network in a café or similar. LXF

QUICK TIP

You can learn more about *udev* here: https://wiki.archlinux.org/title/udev

» BOOTLOADERS AND SYSTEMD

While the Linux kernel is in charge of interacting with the hardware, it is not the only thing required to make up your distro. Plenty of other tools, developed by the GNU project and others, are required. One tool is the bootloader, which loads the basics of the operating system, so everything is then available. The venerable GRUB is often used, which gained popularity over the previous common tool, LILO.

Systemd began life as an init system and was designed to address some of the perceived issues with the well-tested and well-used init *v.* Your init system is the first process loaded and is in charge of managing all other system processes. Distros spent years optimising the boot process to speed it up as much as possible. When designing *Systemd,* techniques such as parallelisation were used and the concept of dependencies was built in to allow the system to boot in as optimised state as possible. For example, starting a service that's reliant on network storage before the network storage has mounted is not a good idea, and this is easy to control with *Systemd*.

The scope of *Systemd* has expanded to cover areas such as the init system, configuration of the networking stack, container management, home directory control and firewall management, and has a boot process stub, called systemd-boot, that loads EFI binaries, such as GRUB.

SUPERFILE

Credit: https://superfile.netlify.app

Fancy file management

While CLI purists scoff at TUI tools, **Shashank Sharma** has no hesitation working with wonderfully crafted utilities that do their job well.

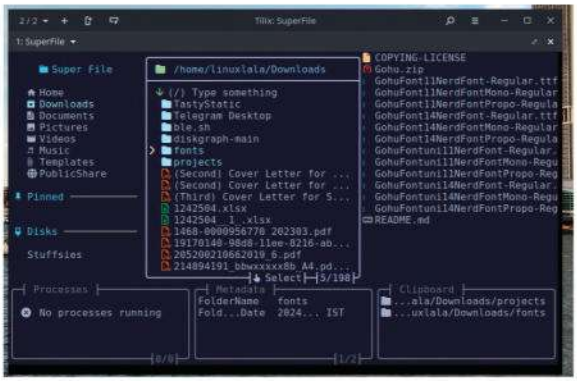
OUR EXPERT

Shashank Sharma is a trial lawyer in New Delhi and an avid Arch user. He's been writing about open source software for 20 years and lawyering for 10.

Although a file manager has little to offer besides the ability to copy, move or delete files and folders, and generally navigate to the directories of your choice to access the files you're looking for, it is one of the most important and frequently overlooked productivity tools. We've already covered a few CLI gems in previous issues, but this month we're talking about *Superfile,* a file manager with the right mix of fancy and modern to make it well worth a look.

Released under the MIT licence, *Superfile* is a configurable file manager that's all about user experience. Unlike most other CLI file managers, *Superfile* has a distinct appearance, using different panels to display useful information. While some modern TUI utilities have incorporated mouse support into their operations, *Superfile* is intended only for keyboard warriors. Thankfully, the project makes it very easy to tweak the keybindings to perform the various operations if you dislike the default offerings.

Feature-wise, *Superfile* supports creating and deleting files and folders, and you can also cut, copy and paste your selection. On top of that, you can quickly view metadata, such as file size, for the



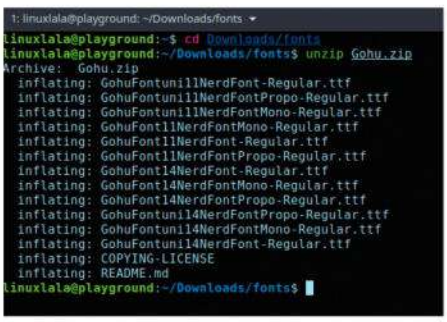
Instead of Browser, the file browser panel shows **S**elect at the bottom when you're in selection mode.

selected file or folder. It also offers a rudimentary search feature, which can help you look for files and folders in the current directory. Unfortunately, however, *Superfile* doesn't as yet support recursive search for files, and the tool also lacks a fuzzy search, something that's commonplace for most utilities nowadays. Both of these features, and many more, have already been requested by the user community, and the project,

INSTALLING A NERD FONT

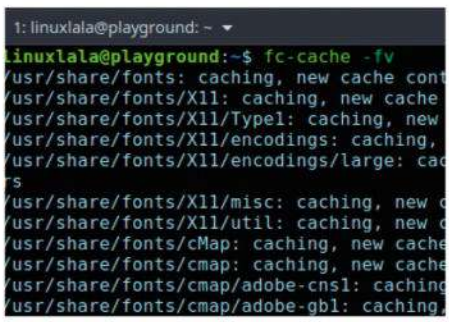
1 Download a Nerd Font

Head over to Nerd Fonts (**www.nerdfonts.com/font-downloads**) and download any font that suits your fancy. Press the Download button under the font of your choice and save the ZIP file to your preferred location. We opted for GohuFont.



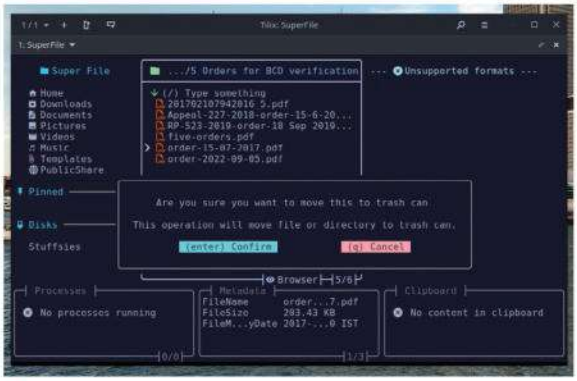
2 Extract the fonts

From the terminal, navigate to the directory where you saved the font's ZIP file. You must extract the fonts with the unzip command: `unzip Gohu.zip`. This extracts a number of TTF font files in the current directory.



3 Install the fonts

Copy the extracted TTF font files to the **~/.local/share/fonts** directory, or to the **/usr/share/fonts** directory if you wish to install them globally. Then run `fc-cache -fv` to refresh the fonts database, which completes the installation.



The project is in active development and constantly adding useful features, such as the one seeking confirmation before deleting files.

which is under active development, may well offer them in time.

Get going

While you won't find this Go utility in the repos of popular distros, the project offers an installation script that you can execute to install *Superfile* easily.

The project readily admits that its goal is to look good. To that end, it relies on Nerd Fonts instead of the standard array of fonts on offer with your distro of choice. This isn't a strict dependency and you can still install and use *Superfile* without installing a Nerd Font, but the project recommends installing one for the optimum display *(see walkthrough)*.

With the font installed, you can run the install script with `bash -c "$(wget -qO- https://superfile.netlify.app/install.sh)"`. This downloads the **install.sh** script and executes it. If your system is already configured for *Homebrew*, you can install *Superfile* with `brew install superfile`. You can now launch *Superfile* with `spf`.

When you first run *Superfile*, you're greeted with a welcome screen that points you to a helpful tutorial on the project's website, reminds you that you can press `?` to view the configured keybindings and thanks you for using the project.

Press Enter to close the welcome screen and proceed to the *Superfile* interface, which features a sidebar on the left that can be used to navigate to common directories such as **Home**, **Desktop**, **Documents**, **Pictures** and so on, as well as mounted partitions. Unlike many other CLI and TUI tools, *Superfile* doesn't adjust the display to fit the terminal emulator window, so you might wish to resize your terminal emulator screen or switch to full-screen.

The middle of the interface features the file browser panel. To its right is the file viewer panel, which shows the contents of the selected directory in the file browser panel, or the contents of the selected file, if *Superfile* supports the format. For now, the file viewer panel only shows the contents of log and text files. In our tests, the file preview rendered a highly pixellated version of image files.

At the bottom are three additional panels. On the left is the Processes panel. Should you choose to paste, move or delete files or directories, those operations and their progress is reported here. To its right is the Metadata panel, which shows the name and size of the selected file or directory. When you copy files or directories, these are listed in the Clipboard panel on the bottom-right of the interface.

You'll figure out the keybindings with time, but here are some of the most useful ones to get you started.

Useful keybindings

Keybinding	Function
?	View the configured keybindings
s	Switch focus to the sidebar
m	Switch focus to the metadata panel
p	Switch focus to the processes panel
Up arrow or **k**	Move up
Down arrow or **j**	Move down
f	Toggle file preview panel
n	Create new file panel
Tab	Switch between the file panels
w	Close the selected file panel
v	Switch between normal and selection mode
J	Select down in selection mode
K	Select up in selection mode
Ctrl+c	Copy selected files and directories
Ctrl+v	Paste selected files and directories
Ctrl+d	Delete selected files and directories
.	Toggle display of hidden files
/	Access the search bar

As with most command-line utilities, *Superfile* is case-sensitive. It also supports extracting contents from a compressed file, or creating a ZIP file from the selected file. When creating a new file, press Ctrl+n, which opens a dialog box where you can type the name of the file. Add a trailing slash (`/`) to the end of the name if you wish to create a directory instead.

Make changes

If you dislike the default keybindings, you can tweak them by editing the **~/.config/superfile.hotkeys.toml** file in your favourite text editor. You can also change the appearance of *Superfile* by switching to a different theme. Open the **~/.config/superfile/config.toml** file in your favourite editor and change the `theme = 'catppuccin'` line. Head over to the theme list page on *Superfile's* website (**https://superfile.netlify.app/list/theme-list/**) to view the different themes on offer. Save the file after changing the theme name and launch *Superfile* afresh to view the new theme in action.

The *Superfile* file manager is a work in progress, with several new features and enhancements already requested by its users. We'll keep a close eye on the project to see how it shapes up, but for now, it remains a nifty little tool that's both fast and pretty. LXF

QUICK TIP

Don't despair if you delete a file or directory accidentally. You'll find these in the ~/.local/share/Trash directory.

LINUX BASICS

Installing and managing software

Part Three! Don't miss next issue, subscribe on page 16!

From app stores to adding package archives and wrestling with sandbox formats, **Nick Peers** runs through the software options.

OUR EXPERT

Nick Peers has been left scratching his head as he runs through all the different methods of installing software. Thankfully, that leaves his other hand free to keep count.

When it comes to installing software on your new Linux machine, you're spoiled for choice. And we don't just mean by the sheer variety of applications, games and tools on offer. In many cases, you have a choice of how you want to install those programs, too. Prepare to be bedazzled (and confuddled) by such terms as 'Apt', 'DEB packages', 'Snap', 'Flatpak' and 'AppImages', to name but some of the many installation methods available to you.

Each of these methods works in a different way, as you'll see later. But let's start with the most obvious source for new software: your distro's package manager. This provides you with an easy way to find and install programs without having to use the terminal, even though last month's tutorial hopefully revealed that it's not as scary or complex as you might have thought.

The package manager varies from distro to distro. Ubuntu 24.04 LTS users will find *App Center* prominently displayed on the Dash, while Linux Mint users can open *Software Manager* directly from its Start menu shortcut. Once open, you'll see the tool provides a reasonably intuitive interface for you to use.

The annotation *(opposite page)* reveals how this works in Ubuntu, but other distros – including *Software Manager* – work in a similar way. You basically browse the applications on offer or search for a specific package, review the versions offered, select one and click Install. The application is installed, and shortcuts are placed in the appropriate places for easy access.

Mint's Software Manager works in a similar way to Ubuntu's App Center, but Mint chooses to work with container-based apps from Flatpak rather than Snap.

Beneath the hood

While you might think that package managers source their software from the same place, sadly this isn't

» AN ALTERNATIVE PACKAGE MANAGER

Whisper it, but the *App Center* isn't the perfect tool for managing your PC's software collection. Not only does it prioritise Canonical's Snaps over its own repositories, but it's rather limited, too.

The *Apt* command-line package manager is far more powerful and versatile, but not everyone wants to learn its intricacies. This is where *Synaptic* comes in. Not only does it work exclusively with Ubuntu's traditional *Apt*-based repositories, but it also provides features not found in the *App Center* (or Mint's *Software Manager*), such as a Lock Version option under the Package menu. This enables you to ignore all updates for that package going forward for whatever reason.

Synaptic can run alongside *App Center* or *Software Manager* – indeed, it's pre-installed on Mint machines. If you're running Ubuntu, install it through *App Center* or via the terminal with `sudo apt install synaptic`.

Once in place, open the tool – it's not the sleekest or most user-friendly, but it's quick and responsive, and won't take you long to grasp the basics. Simply browse or search for packages, then click the checkbox next to them, choosing Mark For Installation. *Synaptic* automatically marks any dependencies for installation, too, making it clear what's required for that package to run. Another advantage of *Synaptic* is that you can mark multiple packages to install in one go. Once configured, just click Apply to install the lot.

QUICK TIP

Wondering if a package exists in the official Ubuntu repos? Visit https://packages.ubuntu.com, where you can search by keyword for releases for a specific release, such as noble or noble-backports.

true. In the case of Linux Mint, its *Software Manager* gives you a choice between System Package and Flatpak. Ubuntu's *App Center* is now focused on delivering apps via Snap, although it does support – in a roundabout fashion – older 'system packages' like those offered by Mint. So, what's the difference between these different sources, and which one should you choose?

Let's start with basic system packages. These come from the most traditional source of software for all Debian-based distros like Ubuntu and Mint: special archives known as repositories, or repos. They can be installed through the terminal using the *Apt* package tool, which we covered in last month's *Linux Basics* tutorial. Traditionally, package managers such as *App Center* were designed to provide more user-friendly access to these packages through a neat graphical front-end.

There's a major drawback to packages hosted in repositories. First, those packages hosted in system repos aren't updated as regularly as the packages themselves. Due to the effort involved, they're mostly frozen at the point of each major system release. This means you can be left waiting for six months or even two years for a newer version to be released. That's fine if the app works and you have no need for new features, but what happens if you want those new features, or you stumble upon a bug?

Apt workarounds

There are two primary workarounds for this problem. The first is offered by Ubuntu itself in the form of a special backports repository. Visit **https://help.ubuntu.com/community/UbuntuBackports** for details. By default, backports are installed manually through the Terminal using the following syntax:

```
sudo apt install <package>/<release>-backports
```

Substitute **<package>** with the name of the package you'd like to install or update, and **<release>** with your current Ubuntu or Mint release name, such as **noble** for Ubuntu 24.04. For example, the following installs the latest version of the *Cockpit* web interface:

```
$ sudo apt install cockpit/noble-backports
```

If you don't know your current release name, type:

```
$ lsb_release -cs
```

Other distros have their own equivalent of backports – for example, Debian offers both backports and testing repos, the latter being based on what's set to be included in the next Debian release.

Developer PPAs

Backports are far from comprehensive, which is why some developers provide their own dedicated repos to provide access to later (if not the latest) versions of their software. These are known as Personal Package Archives, or PPAs for short. By adding these unofficial repositories alongside your distro's existing repos, you allow the *Apt* package tool to search these for software, too – including later versions of apps found in the official repos.

Things can be complicated by the need in some cases for developers to maintain multiple PPAs to cover different distros as well as different versions of distros, such as Ubuntu 23.10 and Ubuntu 24.04. This is due to the different requirements for running that package. But if a PPA exists that supports your distro (and version), you can enjoy more frequent updates.

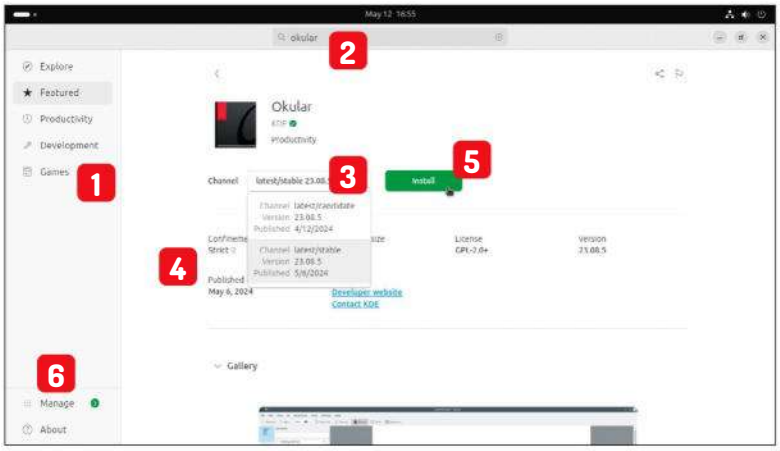
Adding these repos to your system requires a bit of wizardry, but not too much. The step-by-step guide *(over the page)* reveals how to add third-party repositories to your system and manage them going forward. Before doing so, be warned: third-party repos aren't guaranteed to be safe, secure or stable.

If any of this puts you off, keep reading. There's a much easier approach to installing and keeping software updated.

Run in a sandbox

One of the reasons your distro's *Apt* repositories don't provide regular updates of applications is because they can be fiddly to maintain. Many programs you install rely on other packages – known as dependencies – and it's quite common for certain applications to rely

NAVIGATE UBUNTU'S APP CENTER



1 Navigation
You can quickly see what apps are highlighted and on offer using these categories.

2 Find apps
Use the search tool to find an app by name – you may need to choose between Snap and package versions.

3 Select channel
Some apps provide stable and testing (beta) versions. Choose which version to install here.

4 App details
Find out more about the app here – confinement is a security setting used by Snaps to restrict app access.

5 Install app
Click here to install the selected version of the app – you're prompted for your user password.

6 Manage updates
Click here to see what updated packages – including system Snaps – are available.

QUICK TIP

When updating packages, you may come across problems that result in error messages or packages being held back indefinitely. Try `sudo apt --fix-missing update` followed by `sudo apt install -f` in the terminal as a possible fix.

```
nick@nick-ubuntu2404:~$ sudo apt install cockpit/noble-backports
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Selected version '316-1~bpo24.04.1' (Ubuntu:24.04/noble-backports [all]) for 'cockpit'
Selected version '316-1~bpo24.04.1' (Ubuntu:24.04/noble-backports [amd64]) for 'cockpit-bridge' becau
Selected version '316-1~bpo24.04.1' (Ubuntu:24.04/noble-backports [amd64]) for 'cockpit-ws' because c
Selected version '316-1~bpo24.04.1' (Ubuntu:24.04/noble-backports [all]) for 'cockpit-system' because
Selected version '316-1~bpo24.04.1' (Ubuntu:24.04/noble-backports [all]) for 'cockpit-storaged' becau
Selected version '316-1~bpo24.04.1' (Ubuntu:24.04/noble-backports [all]) for 'cockpit-networkmanager'
Selected version '316-1~bpo24.04.1' (Ubuntu:24.04/noble-backports [all]) for 'cockpit-packagekit' bec
The following additional packages will be installed:
  cockpit-bridge cockpit-networkmanager cockpit-packagekit cockpit-storaged
  cockpit-system cockpit-ws
Suggested packages:
  cockpit-doc cockpit-pcp cockpit-sosreport udisks2-btrfs udisks2-lvm2 mdadm
  sssd-dbus
The following packages will be upgraded:
  cockpit cockpit-bridge cockpit-networkmanager cockpit-packagekit
  cockpit-storaged cockpit-system cockpit-ws
7 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 7,138 kB of archives.
After this operation, 15.4 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Backports make it possible to ensure selected apps available from Ubuntu's repositories can be updated to newer versions so they're no longer months – or even years – out of date.

on specific versions of other packages, too. Throw in additional requirements depending on which Linux distro you're running, and things can get really complicated, and not just for the app's developers and maintainers. For example, what happens when you discover two or more applications on your system require different versions of the same package?

The solution lies in containerising apps. This isolates apps inside their own virtual environment, known as a sandbox. The sandbox can be filled with the exact version of any dependencies required by that specific application, which allows it to run freely without affecting anything else on your system.

The upsides are convenience, compatibility, stability and even security. This applies to app development as well as maintenance – once an app is updated, it's accessible to all distros. The downside is that a containerised app takes up more space – significantly more in some cases – than a regular app installed via *Apt*. That's because each container contains all the specific dependencies required to run that app.

Snap to it

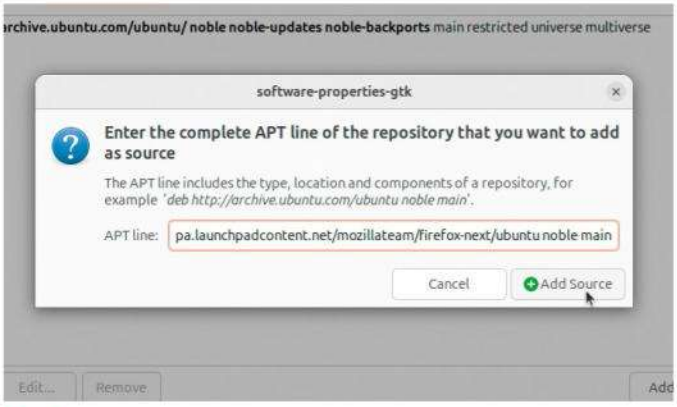
In a simpler world, there'd be a single sandbox system, but instead we have two competing standards. On the one hand, there's Snap (**https://snapcraft.io**), which is developed by Ubuntu's parent Canonical, and so comes pre-installed with Ubuntu. On the other, there's Flatpak (**www.flatpak.org**), which is favoured by most other distros, including Mint.

There are pros and cons to both – for example, Flatpak only works with desktop apps, while Snap can

ADD THIRD-PARTY REPOS AND PPAS

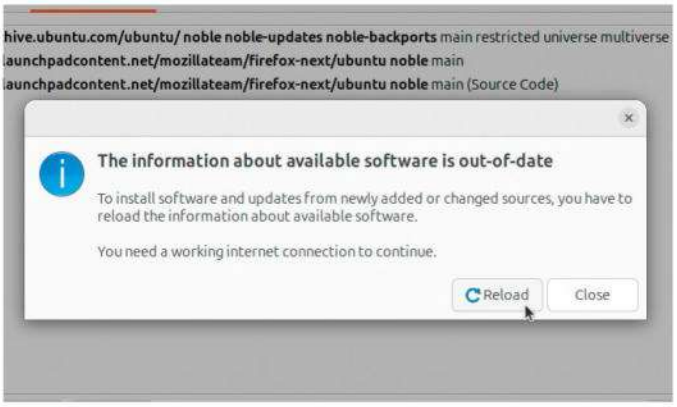
1 Locate repo information

If you're lucky, full instructions for adding the repo to your system will be provided by the application's website, such as **www.openshot.org/ppa/** for *OpenShot*. Not all apps have a website, though, in which case try a dedicated search tool such as **UbuntuUpdates.org** (**www.ubuntuupdates.org/ppas**) or Canonical's own service at **https://launchpad.net/ubuntu/+ppas**, where many third-party PPAs can be found.



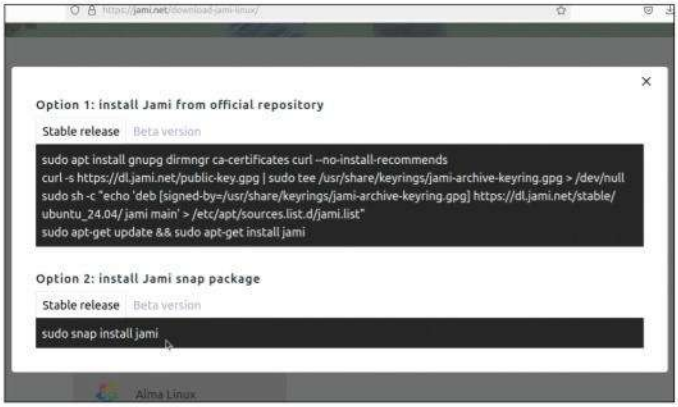
2 Add PPA from Launchpad

If the application has its own *Launchpad* PPA, expand Technical Details About This PPA and select Noble (24.04) from the drop-down menu. Copy the first line beginning `deb` and ending `/ubuntu noble main` to the clipboard. Now open Software & Updates > Other Software tab from the Launcher, click Add... and then paste the line into the APT Line box. Click Add Source .



3 Verify new entry

After being prompted for your password, the entry should appear in the Other Software list. Make sure it's ticked (it may add other entries, but these can be ignored). Now click Close and choose Reload to perform an 'Apt update'. You should now be able to install the app or get a newer version through *Apt* or the *App Center*.



4 Add third-party repositories

If a third party hosts their repo outside of *Launchpad,* you need to perform additional steps, which are best done in the terminal following the instructions on the package's own site or on **UbuntuUpdates.org**. It typically involves two commands: one downloads a key to 'sign' the unofficial repo, the other then adds the repo itself.

also be used for command-line packages. Flatpak is completely open source, while parts of Snap are closed off. Flatpak applications are known to start a bit quicker than Snap ones. But the good news is that it's not an either-or situation – you can have both installed side by side and then decide which one to use on a case-by-case basis.

Add Snap to Mint

Installing Snap can be a little tricky if you're a Mint user. First, Mint has configured a block to prevent Snap being installed. If you're determined to install Snap alongside Flatpak, start by issuing the following command in a Terminal window:

```
$ sudo rm /etc/apt/preferences.d/nosnap.pref
```

From here, installing Snap is the same as it would be for other Debian-based distros:

```
$ sudo apt update
$ sudo apt install snapd
$ sudo snap install core
```

You can now install Snaps from the command line, simply with `sudo snap install appname`. If you'd prefer the convenience of a store app, add the following line:

```
$ sudo snap install snap-store
```

Install DEB files

Sandboxes and system repos aren't the only way to install new software. Some applications provide their own self-contained installers in the form of DEB and/or RPM files. These are basically identical to the system packages you install through your distro's repositories, except they're installed manually. Debian-based distros should choose the DEB file because it works with *Apt* – RPM files require the *Yum* package manager. It works in a similar way – just substitute `yum` for `apt`.

Installation is simple – simply download the DEB file to a suitable folder through your web browser, and then double-click the file and follow the prompts (Ubuntu users should choose Software Install if prompted). You should see it open in a dedicated window. Read the description – if there is one – and then click Install Package.

Once installed, the application should add its shortcut to your launcher or Start menu, plus you can manage and remove it through the main package manager. Applications installed through DEB files don't automatically update, so you need to check the main website regularly for updates, then download and install the newer version over the top of the old one.

Run portable apps

Applications can also be packaged as standalone AppImages. These single files with an .appimage extension contain everything needed to run the app. No installation is required, just double-click the .appimage file each time you want to run it.

Before launching any AppImage, you first need to verify that it has the correct permissions required to let it run. Right-click the .appimage file in Files and choose Properties. Ubuntu users should ensure the Executable As Program switch is flicked to On, while Mint users need to switch to the Permissions tab and tick the Allow Executing File As Program box before clicking OK.

AppImages are self-contained, but store user data elsewhere on your system, enabling you to update the AppImage when a newer version comes out without losing your settings. This location varies, but most choose to store their settings in a hidden subfolder inside your personal **Home** folder. Open Files and press Ctrl+H to view hidden items, where you'll likely track down your app's settings inside the **.config** folder.

The major downside of AppImages is the lack of a convenient shortcut to the program – you can fix this by installing *AppImageLauncher,* as outlined in the box *(above).* Find out more about AppImages at **https://appimage.org** and visit **https://appimage.github.io** to access a catalogue of no fewer than 1,420 apps that are available in .appimage form. LXF

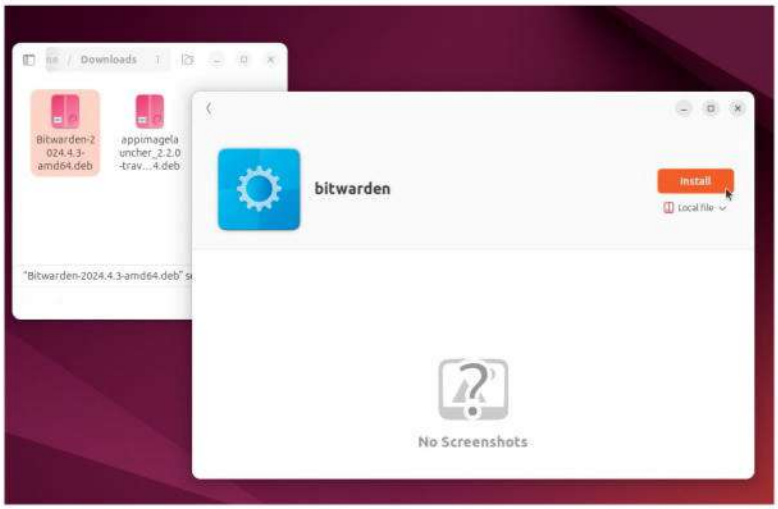
INTEGRATE APPIMAGES

Just because AppImages are portable doesn't mean you can't incorporate them into your system like regularly installed apps. Thanks to *AppImageLauncher* you can consolidate them all into a single folder, plus provide convenient shortcuts.

AppImageLauncher may not have been updated in years, but it works fine in Ubuntu 24.04 and Mint, so long as you install the .DEB version. Navigate to **https://github.com/TheAssassin/AppImageLauncher/releases/** and download the **bionic_amd64.deb** release.

Once installed, *AppImageLauncher* waits silently for you to download and double-click your first AppImage file. It then launches its first-run wizard, where you'll see it offer to create a single parent folder (**Applications**) inside your **home** folder for all AppImages to reside (click Customize if you want to change this). Click OK.

You'll then see a dialog that appears each time you open an AppImage file for the first time going forward. It gives you the choice of simply running the AppImage or choosing to Integrate And Run. This latter option is the one that moves the file to your **Applications** folder and creates a convenient shortcut for easy access. Ubuntu users will find these under Launcher, and Mint should find the shortcuts being placed logically on the Start menu.



Think of DEB files as traditional program installers. Unlike Snaps, Flatpak or packages installed through repos, they don't update automatically.

FLATPAK

Credit: www.flatpak.org

Install and control Flatpak software

Get installing tools with the distro-agnostic packaging system that offers all sorts of advantages. **Michael Reed** has come packing his hex key set.

OUR EXPERT

Michael Reed managed to assemble his work desk from an Ikea flat-pack, so we knew he would be up to the task of creating a Flatpak tutorial.

Flatpak is an application distribution (aka packaging) system for Linux that can retrieve and install software from an online repository or from downloadable package description files. This might make it sound like the standard packaging systems that are central to every Linux distribution, but it has a number of advantages over those traditional systems.

First off, built-in packaging archives can be just a snapshot archive of software, so distribution packages can often be out of date. Flatpaks were built to enable time-poor developers to easily knock out the latest builds of their software. When a developer packages their application with Flatpak, they make a single package that can run on all of the major Linux distributions. It's also appealing because the developer can include all needed dependencies and libraries with the application. This gives a further advantage because it means that you are running the tested and approved combination of components, even if it means that the Flatpak package is running different components on the rest of your system.

These dependencies are bundled into what Flathub calls runtimes, and these are downloaded with each application. This uses a bit of extra space, but Flatpak carries out deduplication, and a runtime that is shared between more than one Flatpak package doesn't have to be installed more than once.

The reason that Flatpak can operate in this way is because it uses what is called sandboxing, which means that each application is isolated from the operating system and the underlying hardware of the computer. Obviously, a piece of software isn't much use if it can't communicate with the resources of the computer and with the user, so it is up to the developer to grant access to these lines of communication – however, this can be overridden by the user. This approach adds to the security and stability of the system. With the right balance of permissions, the application should still be able to operate fully and also integrate with the system in areas such as being listed on application menus.

```
Name     Description                          Application ID       Versio
GNU Im…  Create images and edit photog…       org.gimp.GIMP        2.10.3
GIMP U…  GIMP User Manual                     ….gimp.GIMP.Manual   2.10
Resynt…  Resynthesizer GIMP Plugin            …gin.Resynthesizer   2.0.3
GimpLe…  Lensfun Gimp plugin                  …MP.Plugin.Lensfun   0.2.4
Fourier  Fourier GIMP Plugin                  …MP.Plugin.Fourier   0.4.3
BIMP     Batch Image Manipulation Prog…       ….GIMP.Plugin.BIMP   2.6
Resynt…  Set of GIMP plug-ins that hea…      …gin.Resynthesizer   2.0.3
Liquid…  LiquidRescale GIMP Plugin            …gin.LiquidRescale   0.7.2
GimpLe…  GimpLensfun is a Gimp plugin …       …MP.Plugin.Lensfun   0.2.4
Fourier  A simple GIMP plug-in to do f…       …MP.Plugin.Fourier   0.4.3
FocusB…  FocusBlur GIMP Plugin                ….Plugin.FocusBlur   3.2.6
BIMP     Batch Image Manipulation Prog…       ….GIMP.Plugin.BIMP   2.5
Liquid…  LiquidRescale plugin to resiz…      …gin.LiquidRescale   0.7.2
G'MIC    GREYC's Magic for Image Compu…       ….GIMP.Plugin.GMic   3.3.4
G'MIC    GREYC's Magic for Image Compu…       ….GIMP.Plugin.GMic   2.9.6
FocusB…  Focus Blur plug-in crete a bl…       ….Plugin.FocusBlur   3.2.6
Swatch   Color palette manager                org.gabmus.swatch    1.1
Photop…  Advanced image editor                …elop.photopea_app   2.3.6
Scans …  Create small, searchable PDFs…       …ithub.unrud.djpdf   0.5.4
killermike@killermike-All-Series:~$
```

The Flatpak command-line utility abbreviates its output, so it's a good idea to maximise your terminal window when using it.

Construct your Flatpak

Let's install a Flatpak application to get a feel of how the system works. We'll install *GIMP*, the *GNU Image Manipulation Program*. If you've already got it installed, don't worry, as you can install a Flatpak version of an app alongside the version that you installed using the normal system tools. This is another advantage of Flatpak, as it allows you to check out the latest version of an application without fully committing to it.

We'll start by locating the package by typing `flatpak search gimp` to search the Flathub repository, the default Flatpak repository. This often produces a number of lines of output as there are other packages related to *GIMP* within Flathub, such as plugins and utilities. The top line of output from this command shows the header fields, which are: Name, Description, Application ID, Version, Branch, Remotes. The fields are, for the most part, self-explanatory.

Take the Version field. Generally speaking, the app versions within Flathub are newer than those in a typical Linux distro repository. One reason behind this is because the latest version of a program typically relies on the latest versions of the support libraries,

QUICK TIP

If you have multiple versions of an application installed, you can choose which one is considered the default. Use `man flatpak make-current` for more information.

and it is inconvenient to update an entire Linux distro just to support the latest version of an app. In particular, Flatpak is a good way of getting popular, large applications, such as the latest media apps.

Looking at the top line of results produced by the search command:

Name		Description	
GNU Image Manipulation		Create images and edit photographs	
Application ID	Version	Branch	Remotes
org.gimp.GIMP	2.10.38	stable	flathub

That shows us that the latest stable version `2.10.38` is available and that its application ID is `org.gimp.GIMP`. This application ID consists of three fields separated by a full stop. In this case, the org.gimp part refers to the developer of that application, and GIMP is the object name. As often as not, it's possible to refer to objects simply by their name. So, for example, `flatpak install gimp` works to install *GIMP*, but it's better practice to use the full ID (org.gimp.GIMP) to remove the chance of errors.

Flatpak installation

The full, correct command to install the latest stable version of *GIMP* is: `flatpak install org.gimp.GIMP`. By default, an app installed via Flatpak is installed system-wide rather than on a per-user basis. As Flatpak itself has the necessary permissions, you do not usually need to use a command such as `sudo` in front of a Flatpak command, and this should be avoided. Flatpak prompts you if it needs superuser authorisation.

When you run the installation command, you see an indication of which version will be installed. For example, `org.gimp.GIMP/x86_64/stable` specifies that this is a 64 bit build of GIMP and it is the stable version. Confirming that this is what is required causes Flatpak to spit out more information about the installation.

This is an example of typical output at this stage:

```
ipc    network    x11    dri    file access [1]
dbus access [2]    tags [3]

    [1] /tmp, host, xdg-config/GIMP, xdg-config/gtk-3.0,
xdg-run/gvfs, xdg-run/gvfsd
    [2] org.freedesktop.FileManager1, org.gnome.Shell.
Screenshot, org.gtk.vfs.*, org.kde.kwin.Screenshot
    [3] stable
```

The first chunk of information concerns the permissions that the app will be granted in order to communicate with resources outside of the Flatpak sandbox. `ipc` (inter-process communication) has to be added because X11 (the X Window System that most distros use to run the graphical environment) requires it. X11 permission is granted for the same reason.

```
1 [Flatpak Ref]
2 Name=org.gimp.GIMP
3 Branch=beta
4 Title=org.gimp.GIMP from flathub-beta
5 IsRuntime=false
6 Url=https://dl.flathub.org/beta-repo/
7 SuggestRemoteName=flathub-beta
8 GPGKey=mQINBFlD2sABEADsiUZUOYBg1UdDaWkE
9 RuntimeRepo=https://dl.flathub.org/repo
10
```

Opening a .flatpakref file, we discover that it's a fairly simple file that specifies the location of the resources needed to make the application work.

Permissions such as `file access` are more self-explanatory, but there are some extra notes about these references in the square brackets. This tells us *GIMP* can access directories outside of its own app directory, and the selection is sensible, such as **/tmp**.

The application is also granted D-Bus access. D-Bus is a set of protocols for apps to access the features of the desktop environment. In this case, it allows *GIMP* to take screenshots and also access the file chooser of that graphical environment. Obviously, a program such as *GIMP* would be severely hampered if it could not access the user's files when saving and loading.

Typically, the default permission settings strike a balance between security and essential freedoms.

The runtimes

The next set of information presented is the list of runtimes that have to be installed:

ID	Ver.	Op	Remote	Size
1. org.freedesktop.Platform.openh264	2.4.1	i	flathub	976.5 kB
2. org.gnome.Platform.Locale	46	i	flathub	367.5 MB
3. org.gnome.Platform	46	i	flathub	346.8 MB
4. org.gimp.GIMP	stable	i	flathub	131.1 MB

QUICK TIP

Flatpak installation uses partial object name matches, so, `flatpak install nano` lists projects such as ***N**anonote* and ***N**anosaur*. **D**ouble-check, and use the full object **ID**.

» INSTALLING FLATPAK

Many Linux distributions support Flatpak out of the box, but there are a few that don't. In those cases, you have to install Flatpak yourself. Ubuntu is the most obvious example, as Canonical made the decision to remove Flatpak from the default Ubuntu installation in 2023. Distros such as Fedora have embraced Flatpak, and it's supported on a fresh installation without the user having to do anything.

The easiest way to tell if Flatpak is installed is to open a command-line terminal and type `flatpak --version`. If Flatpak is installed, it reports its version number. If it is absent, the Bash shell tells you that there is no such command.

If Flatpak is not present on your system, how do you add it? We'll use Ubuntu as an example, but this is a method that works on all Debian-derived distributions.

In the command terminal, type `sudo apt install flatpak`. This installs Flatpak, but there's a snag, as Flatpak won't be aware of any repositories. Use the command `flatpak remote-add --if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo` to add it. This prompts you for your password. A reboot of the computer is recommended at this point. From here on, Flatpak should work normally.

Currently, Ubuntu's application store, *App Centre*, doesn't support Flatpak at all. Ubuntu users can install *Gnome Software*, along with the Flatpak plugin, to have a store that supports Flatpak.

QUICK TIP

The full format for the Flatpak installation command is `flatpak install [repository] [object ID]`, but there's no need to specify the repository unless you have manually added extra ones.

Org.freedesktop.Platform is one of the most basic runtimes that is normally required by a GUI application. **Org.freedesktop.Platform.openh264** is a runtime extension, an optional component, and it adds support for H.264 video files. Another common sight among the runtimes that you're likely to find installed on your system, **org.gnome.Platform** is a set of essential libraries required by typical Gnome applications.

The display in this case indicates that the runtime sizes are 976.5kB, 367.5MB, 346.8MB and 131.1MB. This adds up to a nearly 850MB, which could be viewed as quite a lot of space, but as we said, other apps will probably need these runtimes at some point, so the space taken up will be part of a shared resource.

Finishing the installation

Finally, it is time to carry out the actual installation of *GIMP* by accepting the options by pressing Y.

The progress of the various downloads are shown as they are taking place. Once the installation has completed, test that *GIMP* is actually on your system by trying to access it on the launch menu of your desktop environment. The permissions that it was granted make it possible for it to be integrated into the desktop in this way.

There is a slight problem if we have ended up with more than one version of *GIMP*, as it will be listed twice in the app launcher of most desktop environments without any clear indication of which version is which. It would be nice if Flatpak could come up with a simple solution to this. What can work is to left-click on the app launcher icon to see what the launch command is. If it mentions Flatpak, it must be the Flatpak version. If you drag this icon into the desktop or a quick launch bar of your DE, this will be the version that is launched.

Another way of launching the Flatpak version of an app, in the case of multiple versions, is to run it from the command line with `flatpak run org.gimp.GIMP`. This runs the version that was installed using Flatpak.

Removing a Flatpak version of an app is as simple as installing it. Use `flatpak remove [application ID]`. This can, sometimes, leave unneeded runtimes installed. You can list the entire set of runtimes on your system with `flatpak list --runtime`. Remove runtimes that are not in use by any app with `flatpak uninstall --unused`.

Easy updates

It's easy to update all of the Flatpak packages on the system with a single command (`flatpak update`), but there are differences between Flatpak and other Linux packaging systems that are worth being aware of. Unlike Snap, Flatpak doesn't automatically update the packages on your system, so you must do it manually. Another difference is that doing a system update (`sudo apt update && sudo apt upgrade` on a Debian-derived system) doesn't update the Flatpak packages.

It's worth keeping Flatpak runtimes and apps up to date. For example, we ran into a problem where Nvidia hardware acceleration wouldn't work within screen capture application *OBS*. On closer examination, it turned out that we'd forgotten to do a Flatpak update in a while, and the result was that the system Nvidia driver, the Flatpak Nvidia runtime and the Flatpak version of *OBS* were out of sync with each other. Closing down *OBS* and running `flatpak update` got things working as expected once we'd restarted *OBS*.

Manual installation

Going back to the subject of installing *GIMP* as an example application, we might want to take advantage of Flatpak's ability to install multiple versions of an application to check out the unstable version of the forthcoming *GIMP 3*. We can do so without disturbing the stable version, whether that was installed using the system package manager or through Flatpak.

We need to install a .flatpakref file of the latest unstable version. This is a small file that contains the instructions needed to download a specific version of an app. The *GIMP* project provides two versions of the version 3 preview, the development version and the nightly build, described as unstable and very unstable respectively. You can find the .flatpakref downloads on the *GIMP* website (**www.gimp.org/downloads/devel/**).

We can download the .flatpakref file and install it with the following command:

```
$ flatpak install org.gimp.GIMP.flatpakref
```

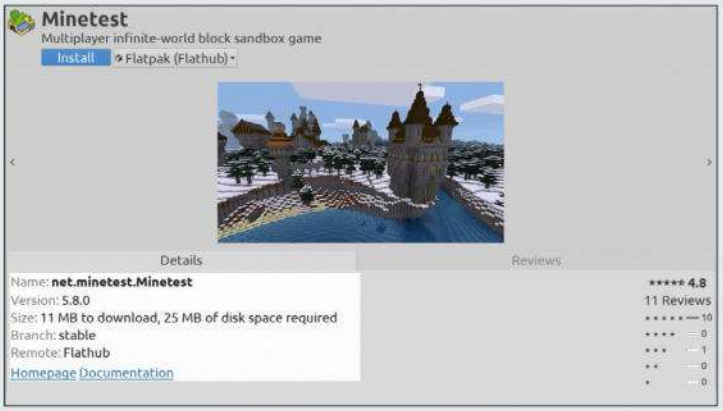
For that matter, it could also have been installed from the URL:

» FLATPAK GUI OPTIONS

Flatpak support is present in some distributions through their GUI software manager. For example, Fedora versions that use *Gnome Software* as their software manager should support Flatpak. This means that the software manager connects to Flathub and offers that version alongside the version in the distro's own repository. If you do come across a setup that doesn't support Flatpak/Flathub, it is sometimes possible to add a plugin to *Gnome Software* to add support. Search the distro documentation to find out how to do it.

Sometimes, a distribution has another software manager that works in the same way as *Gnome Software,* and that often supports Flatpak. One would hope that simply clicking on a **.flatpakref** file in the file manager would launch the correct tool to complete the installation, but we have found support for this be spotty across different distributions.

Warehouse is a front-end designed solely to manage your installed Flatpak applications and runtimes. Add it to your system with `flatpak install io.github.flattool.Warehouse`. *Warehouse* gives a complete overview of the Flatpak applications that are installed on your computer and provides a graphical method to add, remove and tweak the settings of installed applications.



Browsing Flatpaks using **L**inux **M**int's **S**oftware **M**anager. **M**inetest 5.8.0 from Flathub version 5.4.1 is available in the distribution repository.

This version of **GIMP** is clearly marked as an unstable preview version. **U**sing Flatpak, we installed it alongside the stable version.

```
$ flatpak install https://flathub.org/beta-repo/
appstream/org.gimp.GIMP.flatpakref
```

We like the way that there is consistency between the way that Flatpak deals with all three installation cases (installation from the repository, from a file or from a URL) with the same command.

There is another way of installing from a **.flatpakref:** by clicking on it in the file manager. However, we have often found that this isn't set up properly in a typical desktop distro. Check out our suggestions for a more reliable method, using a proper Flatpak GUI, in the boxout *(opposite page)*.

You are presented with a series of prompts and questions that are similar to installation from the Flathub repo. The list of runtimes is slightly different from those for a Flathub installation as the developers have been working with unstable beta versions of certain libraries while working on the latest version of *GIMP*. Proceed by accepting their installation.

You can check that it has installed properly by typing `flatpak list`. Sure enough, among the list of application runtimes, we can see two lines for the different branches of *GIMP* that we have installed:

```
GNU Image Manipulatio… org.gimp.GIMP 2.99.18 beta
flathub-beta system
GNU Image Manipulatio… org.gimp.GIMP 2.10.38
stable flathub system
```

As far as Flatpak is concerned, the object ID of the application is the same for both branches – however, the application is run with an extra flag at the command line. From now on, we can run the 3.x preview (labelled as 2.99.18) by typing:

```
$ flatpak --branch=beta run org.gimp.GIMP
```

We can also run the regular, stable version (2.10.38, in this case):

```
$ flatpak --branch=stable run org.gimp.GIMP
```

If you want to remove this beta from your system, use the same command as before:

```
$ flatpak uninstall org.gimp.GIMP
```

When the command is run, you are prompted with a question about which version to remove:

```
Similar installed refs found for 'org.gimp.GIMP':
  1) app/org.gimp.GIMP/x86_64/beta (system)
  2) app/org.gimp.GIMP/x86_64/stable (system)
  3) All of the above
Which do you want to use (0 to abort)? [0-3]:
```

Manage Flatpak permissions

Flatpak's sandboxing mechanism prevents an app from accessing certain resources of the computer and the operating system. Occasionally, you may need to allow an app to access a resource that was unanticipated by the developers, and you do this with `flatpak override`. For example, you can give an application access to a directory with the following command:

```
$ flatpak override [object ID] --filesystem=[directory]
```

Bear in mind that the Flatpak version of an app such as *GIMP* can, by default, access any directory that you can access through the file chooser of your desktop environment. The **--filesystem** option would be useful for an app such as a media server if you wanted to give it access to a directory outside of the **home** directory. This might be a network folder mapped to a mount point within your filesystem such as **/MyMedia**.

On some setups, you might want to restrict the filesystem access. You can forbid a Flatpak-installed application from accessing a particular directory with:

```
$ flatpak override [object ID] --nofilesystem=[directory]
```

Other elements of the system can be exposed to a Flatpak-installed application, but there is a bit of fragmentation in how this is organised. It's unlikely that you would have to give access to hardware and system resources as an end user. Use `flatpak permission-reset` to reset all permissions to their default values for that package.

In summary

Hopefully, we've introduced you to the basics of using Flatpak to install and manage applications. A common approach is to stick with the system package manager (*Apt, RPM* and so on) for normal maintenance of the operating system and use Flatpak to add the latest and greatest when it comes to full applications. Doing this ensures that you have the most up-to-date apps, and it adds a few security and stability advantages thanks to Flatpak's sandboxing features. LXF

Warehouse is a Flatpak front-end to help with installations and configuration of Flatpak installed applications.

BACK ISSUES » MISSED ONE?

ISSUE 316
July 2024

Product code:
LXFDB0316

In the magazine
Transform your desktop with hot KDE Plasma, and join us as we start our journey inside Linux to discover how it works. Plus, choose the best text editor for your writing projects, find out about the phone firm making an environmental impact, build your own weather machine, get back to basics with the Linux terminal, keep your video calls private, and enjoy a wealth of reviews, news and views.

ISSUE 315
June 2024

Product code:
LXFDB0315

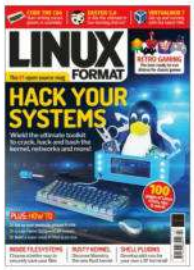
In the magazine
Discover how to upgrade to Ubuntu 24.04 LTS and join us on a journey through the world's most popular distro's 20-year history. And don't miss our *Roundup* of hacker distros, an in-depth look at the impending Epochalypse, detailed tutorials on live-streaming audio via the terminal, setting up Gnome, editing images in *Krita,* and archiving C64 tapes, and plenty more besides.

ISSUE 314
May 2024

Product code:
LXFDB0314

In the magazine
Join us as we explore Linux's infiltration of Windows, and discover how you can also get Windows apps running in Linux as we uncork the latest *Wine* release. And let's not forget a *Roundup* of time trackers, tutorials on vulnerability audits, setting up a home ebook server, streaming games, and more, along with distro reviews, news, and packed Pi and *Adminsteria* sections.

ISSUE 313
April 2024



Product code:
LXFDB0313

In the magazine
Discover how to use the ultimate hacker's toolkit, staying out of trouble while doing so. And join us as we take the Puppy Linux developer's new distro for a run and explore its container features. Plus, we have a *Roundup* of retro-gaming distros, an in-depth look at filesystems, tutorials on adding plug-ins to our **LXF** shell and getting more from your VMs, plus news, reviews and oodles more.

ISSUE 312
March 2024

Product code:
LXFDB0312

In the magazine
Blast off into the future with a look at the five best next-gen distros, and discover whether Raspberry Pi or Orange Pi is the best SBC for you. Plus, learn how to rescue retro media, add NPCs to your own point-and-click adventure, emulate an analogue computer, and lots more. We've also squeezed in hardware and distro reviews, a password manager *Roundup*, news and more.

ISSUE 311
February 2024

Product code:
LXFDB0311

In the magazine
Stay safe online by sending net nasties to *Pi-hole,* and read about the rise and fall (and rise again) of Basic, and the roll-out of the rolling-release Rhino Linux distro. You can also learn how to strengthen your shell history, set up a home CCTV system, add an inventory to your point-and-click adventure, and much more. Plus, check out our packed reviews, news and Raspberry Pi sections.

To order, visit www.magazinesdirect.com

Select **Single Issues** from the tab menu, then select **Linux Format.**

Or call the back issues hotline on **0330 333 1113**
or **+44 (0)330 333 1113** for overseas orders.

Quote the product code shown above and have your credit or debit card details ready.

FYNE AND GO

Credit: https://fyne.io

Part One! Don't miss next issue, subscribe on page 16!

Create a Fyne journaling app

Dressed in his programming finery, **Andrew Williams** shows how you can quickly build a graphical app for your computer using Fyne and Go.

OUR EXPERT

Andrew Williams is a software engineer and entrepreneur based in Scotland. He has been a core developer in open source projects such as Enlightenment, EFL and *Maven*. He founded the *Fyne* toolkit.

Over the past decade, we have seen great improvements in the graphical software user experience and corresponding advancements in the tools that are used to create compelling apps for our phones and tablets. Unfortunately, building graphical applications for the Linux desktop seems not to have benefited quite as much from these developments. With this in mind, several recent projects have been created, aiming to serve all platforms, so none are left behind.

The *Fyne* project is a popular toolkit that was started with this purpose in mind. This cross-platform GUI toolkit for Go enables you to create a single codebase from which graphical apps can be built for Linux, BSD, Android, iOS, Mac OS and Windows. *Fyne* aims to make it easy for developers to build compelling graphical apps for desktop, mobile and web.

In this article, we step through the setup and basic coding to create an application user interface. In the next part, we will add some advanced features and explore packaging and distribution options.

Getting started

Fyne is based on the popular Go programming language. Go is noted for its ease of use and quick compiling. However, because *Fyne* connects to graphics drivers to build high-performance GUI apps, we also need to have other developer tools installed, including a C compiler and some development headers.

The exact steps required to install Go (at least version 1.17), *GCC* and the X11 development tools vary for each platform, but the most popular are as follows:

- **Debian/Ubuntu:**

```
$ sudo apt-get install golang gcc libgl1-mesa-dev xorg-dev
```

- **Fedora:**

```
$ sudo dnf install golang gcc libXcursor-devel libXrandr-devel mesa-libGL-devel libXi-devel libXinerama-devel libXxf86vm-devel
```

- **Arch Linux:**

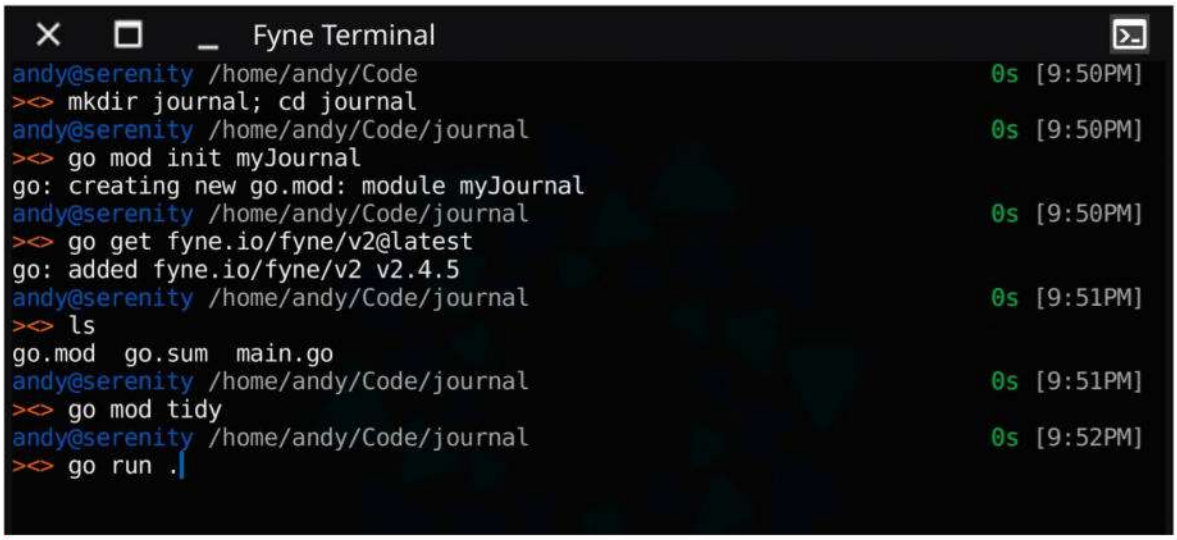
```
$ sudo pacman -S go xorg-server-devel libxcursor libxrandr libxinerama libxi
```

More information and additional distributions are covered at **https://docs.fyne.io/started/**.

With these tools installed, it is possible to get started coding our app. Since Go 1.16, it is standard to create a module to manage dependencies, so we have a few setup steps before we open our code editor. First, it is a good idea to create a new folder for your work, and then we initialise a new module (with a

QUICK TIP

There is a lot of functionality in *Fyne* – it can be difficult to remember everything when getting started. Using an IDE with auto-completion can help a lot, as the API is designed to be consistent and the standardised patterns allow a code editor to make smart suggestions.



The commands we have used to run this project, displayed in Fyne Terminal.

unique identifier for our project). Lastly, we ask Go to fetch the *Fyne* dependency for our project, which it looks up online to find the source code:

```
$ mkdir journal; cd journal
$ go mod init myJournal
$ go get fyne.io/fyne/v2@latest
```

Opening a window

To get started, we open a new file, usually named **main.go**, and start with a package declaration and any imports that are required. We are using three packages from *Fyne*. These are the **root** package, where basic types are defined, the **app** package, which provides the app runtime, and the **widget** package, which contains standard widget implementations.

```
package main

import (
  "fyne.io/fyne/v2"
  "fyne.io/fyne/v2/app"
  "fyne.io/fyne/v2/widget"
)
```

We now enter code that opens our app and user interface. As with any Go application, we create an entry point by defining a **main** function. Inside this, we insert the first lines of *Fyne* code: loading a new app and creating a window. The **app** instance is what sets up the application context and loads any drivers that are required. The **window** variable **w** represents a window on your computer desktop. Passing the **title** parameter to the constructor sets what is displayed at the top of the border, if your window manager displays a title bar:

```
func main() {
  a := app.New()
  w := a.NewWindow("My Journal")

  // more code goes here shortly
}
```

That is most of the code set up. The final thing we need to do is actually set some content on the window. In this example, we will set an **Entry** widget, because this will make the content of our journal application that we are building in this article.

Once the content is set, we show the window and run the application using **ShowAndRun()**:

```
w.SetContent(widget.NewMultiLineEntry())
w.ShowAndRun()
```

With the code all written *(you can see the complete code in the Our First Fyne Application box, above)*, we can run this and see the app window! The following steps check all dependencies are downloaded and then compile and run the app.

Note that the first time you do this, there is a lot of graphics code to compile and it can take minutes rather than seconds – all future builds will be very fast.

```
$ go mod tidy
$ go run .
```

» OUR FIRST FYNE APPLICATION

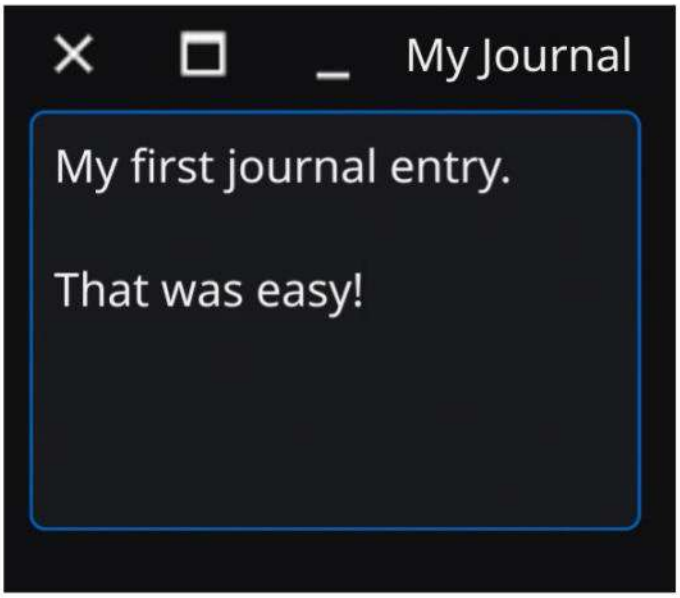
Using just the following lines of code, we can get a window displayed on screen that allows us to type our very first journal entry:

```
package main

import (
    "fyne.io/fyne/v2"
    "fyne.io/fyne/v2/app"
    "fyne.io/fyne/v2/widget"
)

func main() {
    a := app.New()
    w := a.NewWindow("My Journal")

    w.SetContent(widget.NewMultiLineEntry())
    w.Resize(fyne.NewSize(200,150))
    w.ShowAndRun()
}
```

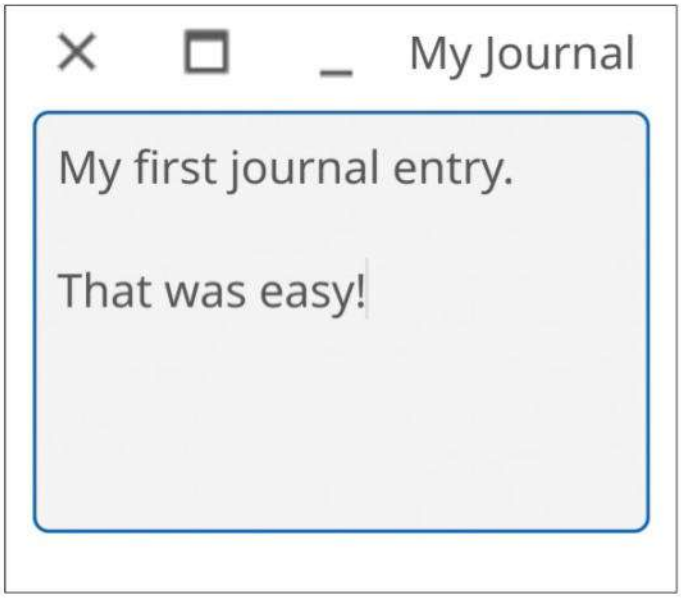
If you have completed the setup steps, the code above can be run by just typing `go run main.go`, and the window appears on screen after it compiles for a few seconds. Now we can add functionality!

QUICK TIP

Go is a compiled language, which means the app built using it will run very fast. Its compiler also runs very quickly, but the first time you compile a *Fyne* app it also needs to build the graphics drivers, which can take a minute or two.



Our window showing an entry under the title bar.



The same window loaded on a computer using a light theme.

QUICK TIP

We are building a native app, so you need the compiler set up and development headers as described in the Getting Started section *(page 64)* – if you see build errors, double-check that you have done these steps first.

With this running, you will see a window that looks like our screenshot *(previous page, bottom-left)*. You can type your text into this input box. When the window is resized, you will see that the content fills the available space.

If your default theme is light mode, you will see an interface like the other image *(previous page, bottom-right)*. *Fyne* attempts to match your user preferences for whether you are in light mode or dark mode. However, you can override this by setting the **FYNE_THEME** environment variable to **light** or **dark**.

Coding our app interface

Now we have the app running, we can look more at the contents of our user interface. The journal app should have a navigation bar at the top showing the current date and previous/next buttons, which move through time. Underneath that, taking up the rest of the window, is a single input widget (much like our first tests) where users can enter their daily notes.

To add all this functionality, we need to set up more widgets and then lay them out in a container. First of all, we will create a **Label** widget alongside the **MultiLineEntry** that we created earlier. **Label** is the standard way to display user interface text that can form navigation or other information display. In this case, we will set a date that will be updated later, and centrally align the text by setting the **Alignment** field to **fyne.TextAlignCenter**.

To complete the new widgets required, we will introduce the buttons that will be used for the back and forward navigation. For this we use **NewButtonWithIcon**, passing an empty string for the label so it is just an icon button. For icons, we can use the **theme** package (your IDE should auto-add the right import), which provides a back and next icon for navigation. This also contains many other icons that you can use in your apps.

The button widget constructor function also takes a callback function parameter – we will fill that in later; for now, we just leave it empty.

```
entry := widget.NewMultiLineEntry()
title := widget.NewLabel("9 May 2024")
title.Alignment = fyne.TextAlignCenter

prev := widget.NewButtonWithIcon("",
  theme.NavigateBackIcon(), func() {})
next := widget.NewButtonWithIcon("",
  theme.NavigateNextIcon(), func() {})
```

With all the widgets created for our app, we need to define how to arrange them.

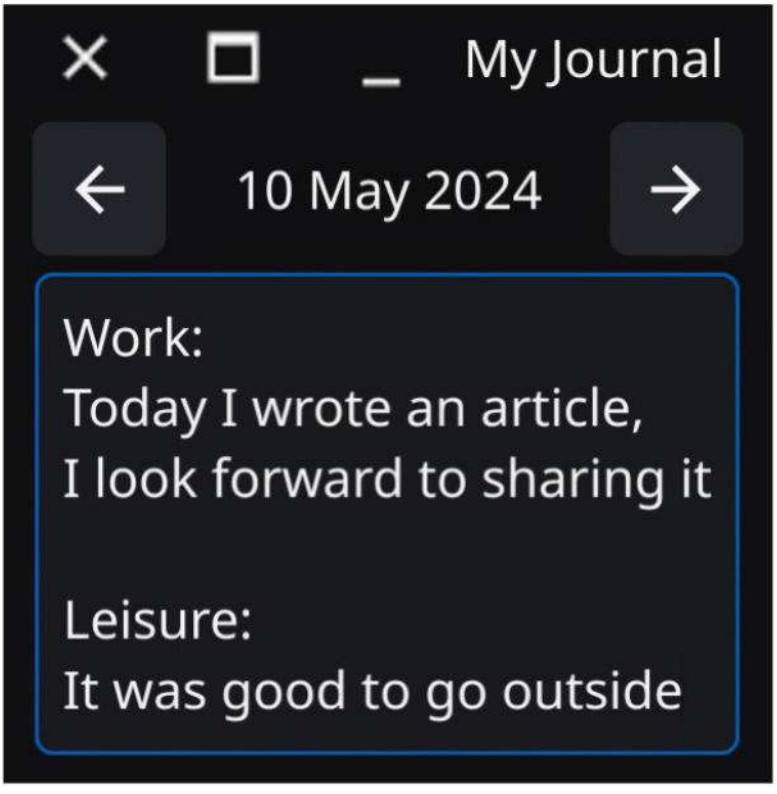
In *Fyne*, this is done with containers. Each container has a layout and the **container** package provides a number of constructor functions to help set up standard arrangements.

In our app, we will make use of the **Border** container, which allows widgets to be placed on the top, bottom, left and right sides of an area, with other items filling the remaining space.

We use two **Border** containers. The first **Border** container is used for the navigation bar that arranges the buttons on either side of the title. The second border container places the overall bar above our main content. The code for these looks as follows:

```
bar := container.NewBorder(nil, nil, prev, next, title)
w.SetContent(
  container.NewBorder(bar, nil, nil, nil, entry))
```

The **SetContent** call above replaces the previous instance; setting the window content a second time would just overwrite it, rather than performing any automatic arrangement or stacking. You can now run the application again with `go run .` and you will see a screen that looks like the screenshot *(left)*.



Displaying the user interface components of our journaling app.

Navigating time

With the interface created, we need to add some functionality to bring it to life. First we will work on some date formatting code so that the navigation bar displays the correct date. Then we will update the buttons to move through time one day per tap. The first thing we will do is set up the date format that we will use. Go defines its own time format system, which you can read more about at **https://pkg.go.dev/time**.

```
const dateFormat = "2 Jan 2006"
```

With the date format set, we add more code into the **main** method to store the current date that is being displayed and then add an inline function called **setDate** that is used to update the various components. When **setDate** is called, it stores the date in case we need it later, formats the string version of this date and then updates our navigation bar title using **SetText**. This helper function is then called immediately with the current date to set up the user interface when it is first loaded.

```
var date time.Time
setDate := func(d time.Time) {
  date = d
  dateStr := date.Format(dateFormat)
  title.SetText(dateStr)
}
setDate(time.Now())
```

Once the hard code has been written, we can update the buttons to set the correct date. We make use of the **setDate** helper again and call it with a modified date by using **Add** and passing a positive or negative day (24 hours). The buttons will look like the following with that updated:

```
prev := widget.NewButtonWithIcon("",
  theme.NavigateBackIcon(), func() {
  setDate(date.Add(time.Hour * -24))
})
next := widget.NewButtonWithIcon("",
  theme.NavigateNextIcon(), func() {
  setDate(date.Add(time.Hour * 24))
})
```

This completes the navigation code. If you run the app, you can see the date change as would be expected. However, to complete this we need to update the content as well.

Remembering user entries

To complete the journal app for part one of this series, we will store the journal entries in memory. To do that we need to create a map of string (date) to string (journal entry), as follows:

```
var entries = make(map[string]string)
```

Now that we have a place to store the data, we need to use it in the app. The first step is to store journal notes. To do this, we make use of the **OnChanged** field of the **Entry** widget. This is executed every time the user updates the content of the entry by typing into it. The new value is then passed in and we store that information in our **entries** map. We make a date string again for the key to use, and then save the notes into the storage:

```
entry.OnChanged = func(s string) {
  dateStr := date.Format(dateFormat)
  entries[dateStr] = s
}
```

To read the information out of this storage, we will do the equivalent read operation. Inside the **setDate** function that we created, we will look up the map and use the information found to pass into the **SetText** method of our main **Entry** widget:

```
setDate := func(d time.Time) {
  ...
  entry.SetText(entries[dateStr])
}
```

That is all of the code for our app. You can see it all put together in the Complete Journal App Code box *(on the right)*. By running this code, you have a complete application running on your computer. You can also execute `go install` to have this placed into your **~/.go/bin/** folder or you could copy it into **/usr/local/bin** as well.

Looking ahead

In this article, we have seen how easy it is to create an app with a window and standard widgets, including text entry and buttons. We have used in-memory storage to make it functional, but it is not really complete yet.

Come back for part two next month, when we will add persistent storage using some more standard *Fyne* APIs as well as exploring data binding for cleaner code. We will also see how the app can be packaged and installed alongside other graphical apps on your computer, sent to your friends or even uploaded to app stores and marketplaces. LXF

COMPLETE JOURNAL APP CODE

```
package main

import (
    "time"

    "fyne.io/fyne/v2"
    "fyne.io/fyne/v2/app"
    "fyne.io/fyne/v2/container"
    "fyne.io/fyne/v2/theme"
    "fyne.io/fyne/v2/widget"
)

const dateFormat = "2 Jan 2006"

var entries = make(map[string]string)

func main() {
    a := app.New()
    w := a.NewWindow("My Journal")
    var date time.Time

    entry := widget.NewMultiLineEntry()
    entry.OnChanged = func(s string) {
        dateStr := date.Format(dateFormat)
        entries[dateStr] = s
    }
    title := widget.NewLabel("Today")
    title.Alignment = fyne.TextAlignCenter

    setDate := func(d time.Time) {
        date = d
        dateStr := date.Format(dateFormat)
        title.SetText(dateStr)
        entry.SetText(entries[dateStr])
    }
    setDate(time.Now())

    prev := widget.NewButtonWithIcon("",
            theme.NavigateBackIcon(), func() {
        setDate(date.Add(time.Hour * -24))
    })
    next := widget.NewButtonWithIcon("",
            theme.NavigateNextIcon(), func() {
        setDate(date.Add(time.Hour * 24))
    })
    bar := container.NewBorder(nil, nil, prev, next, title)

    w.SetContent(
        container.NewBorder(bar, nil, nil, nil, entry))
    w.Resize(fyne.NewSize(250, 220))
    w.ShowAndRun()
}
```

TEKTRONIX 4010

Credit: https://github.com/rricharz/Tek4010

Reliving the birth of the terminal

It looked like a PC but, although it wasn't user-programmable, the display terminal had a vital role in the days of minicomputers, says **Mike Bedford**.

OUR EXPERT

Mike Bedford worked as a software engineer back in the mists of time, writing firmware for a range of computer terminals. These VDUs emulated the DEC VT-100 and its younger siblings, while also adding that all-important colour capability.

As you head back in time past the '70s, you enter the minicomputer era. Part of that era, which is still very much embedded in modern Linux/Unix, is the terminal. Back then, the display terminal acted as the user's interface to the computer. We could even think of it as a first step towards the client-server model of computing.

We're going to take a look at two classic display terminals, one for handling text and another that could also handle graphics. And while we're going to introduce an emulator for one of these terminals, the bare bones of the other is alive and well and integrated into most Linux terminal software, as we'll see.

Hello VDU

Having raised the subject of the progression of mainframes to minicomputers to personal computers, let's look at this in a bit more detail. In particular, let's see how users communicated with each of these types of computer. This is fairly obvious in the case of PCs – the user interface, comprising a keyboard, pointing device and screen, is integral to the computer. Turning to the other end of the spectrum, most mainframe users had no direct communication with the machine. Instead, they provided their input on punched cards that were prepared offline and passed to a computer operator. They'd eventually receive their output from a printer to which only the operators had direct access.

This brings us to the middle ground occupied by the minicomputer. Introduced in the mid-'60s, these machines were intended for use by several users, each of whom would communicate directly with the computer. At first, the user interface was a teletype, a hard-copy terminal that had a keyboard and printed the computer output on to a roll of paper. These were used because they were already available, having been designed for textual communication over landlines. They weren't ideal, though, because they were slow, couldn't display graphics, and used up a lot of trees.

The use of visual display units (VDU) as computer terminals predated the onset of the minicomputer era, having first made their appearance on mainframes, even though they were used mainly by the computer operators. With the birth of minicomputers, though, low-cost VDUs started to become available, and eventually replaced teletypes. Initially, these VDUs were purely non-hard-copy versions of teletypes, so every character was printed to the right of the previously printed character, and every line appeared below the previous line. For this reason, they were nicknamed glass teletypes. Very soon, though, video terminals offering a degree of local intelligence appeared, and some that could also display graphics. This generation of VDU is our subject here.

DEC introduced the VT100 in 1978 and, in so doing, set the standard for text-based video terminals.

CREDIT: Jason Scott, CC BY 2.0, www.flickr.com/photos/54568729@N00/9636183501

A text terminal – DEC VT100

The VT100 was introduced by DEC (Digital Equipment Corporation) in 1978. This was, presumably, intended mainly for use on its PDP-11 minicomputers. However, it became an industry standard and was employed across a much broader range of minicomputers.

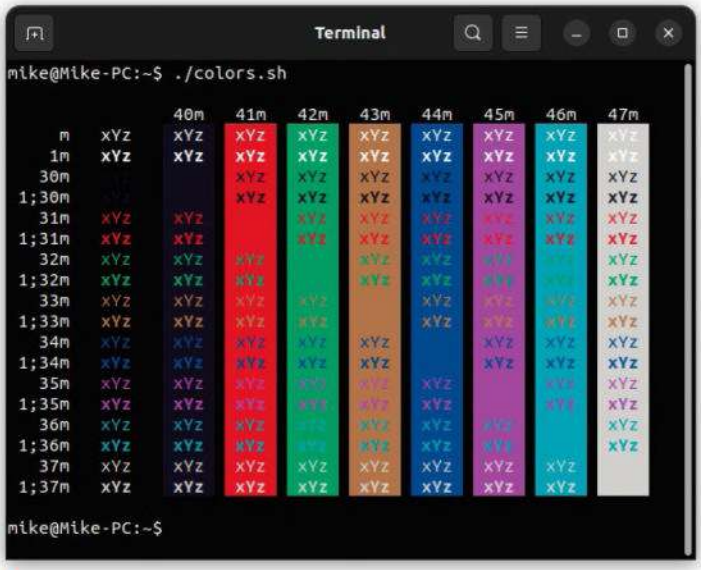
It could display 24 lines of 80- or 132-character text in monochrome – white on a black background. In addition to their default appearance, characters could be displayed in bold, underlined, blinking and/or reversed (black on white). Characters could also be displayed double width or double width and height. Special characters were also available, which were mainly intended for drawing on-screen tables. Perhaps

most importantly, though – and representing the main benefit over teletypes and glass teletypes – was a means of writing anywhere on the screen, not just at the next position. This was achieved via a range of cursor control commands, and augmented with various means of selectively erasing portions of the screen. And, while most users wouldn't have known or cared, the VT100's intelligence was provided by an Intel 8080 microprocessor. VT100s cost around $1,700 which sounds like a lot, even without taking 46 years of inflation into account. However, don't forget that computers like DEC PDP-11s, to which they'd be attached, cost several tens of thousands, and even teletypes cost half as much as a VT100.

The characters and control functions transmitted to the VT100 took various forms. Obviously, there were the printable characters, which had ASCII values from 32 to 7E hexadecimal. Then there were those non-printing ASCII characters referred to as control characters. Included here, for example, are `CR`, which moves the cursor to the first column in the current line, and `LF`, which (depending on a related option) moves the cursor to the same column in the next line. However, because of the number of control functions designed into the VT100, there weren't nearly enough control characters available. Thus most commands took the form of escape sequences, which are commands that comprise the `ESC` (escape) control character followed by one or more ordinary printable characters, which didn't actually print. Take, for example, `ESC # 6`, in which we've only shown the spaces to improve readability, but which shouldn't actually be present. This escape sequence doesn't cause the `#` or `6` to be printed, but results in all the characters on the current line to be rendered at double width.

Although we referred to the VT100's multi-character commands as escape sequences, that term isn't strictly true for all of them. Several of the sequences start with `ESC` and then `[`, but these are control sequences, not escape sequences. This is because they are defined as starting with a character called `CSI` (Command Sequence Introducer), which has the ASCII code 9B hexadecimal. However, in the early days of minicomputers, communication was often via a 7-bit serial interface, so ANSII characters greater than 7F hexadecimal were not available. So, to represent `CSI` in a 7-bit world, `ESC [` was used instead of CSI.

Dubbed glass teletypes, the first VDUs offered little more functionality than hardcopy terminals like teletypes.



Most Linux terminal applications support ANSI X3.64 commands. As a result, they come close to emulating the VT100 and its successors.

Usually, when we look at archaic computers, we recommend emulators so you can try them yourself. We're not going to do that for the VT100, though, because the majority of Linux terminal applications emulate many of the key features of the VT100. The upshot of this is that you can get a feel for what it was like to write code for a VT100 as a historical curiosity, but you can also use this functionality for genuine applications. And while Linux terminals don't emulate quite all the VT100's features, they support most of the really useful ones, and in other ways they go beyond the VT100's capabilities. In particular, they emulate some of the successors to the VT100, which added extra features, and they also support colour, which was absent on pretty much all the VT-series text-only terminals. Colour is provided by implementing additional commands from the ANSI

QUICK TIP

Some terminals allowed printers to be attached to them to reduce the number of connections that were required to the computer. Escape sequences allowed the VDU to be bypassed, thereby simply forwarding what they received to the printer. The printer could also be used to print the screen locally.

» BLOCK-MODE TERMINALS

DEC and Tektronix weren't the only manufacturers of VDUs in the '70s and '80s, such was the appetite for these devices. Many of these other products were clones of the VT100 or the 4100 and similar, but other manufacturers defined their own command sets.

While lots of these terminals operated in much the same way as the VT100 and the 4010, albeit sometimes with non-compatible instruction sets, others did something different. Indeed, one of the DEC VT-series terminals came into this category: the VT132, which made its appearance just a year after the VT100. Its claim to fame was its block mode of operation, which very much reduced the amount of communication with the computer. Let's see how this worked.

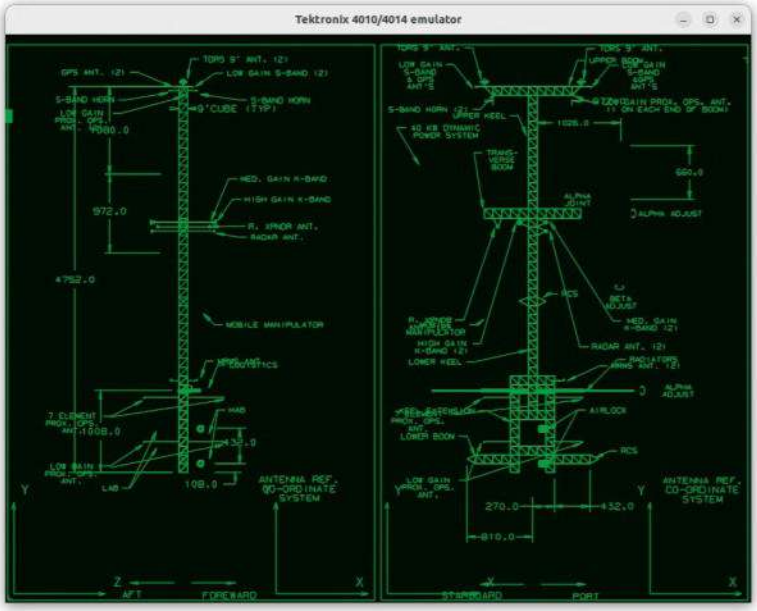
First, data was transmitted to the terminal to define an on-screen form, in much the same way as with a VT100, but here the similarity ends. All the screen was then defined as protected, except for those fields where the user would enter data. The terminal was then put into edit mode. This allowed the user to fill in the form and even edit it by making additions or deletions, but nothing was sent to the computer. When the form was complete, the user hit Enter and the terminal transmitted all the user data as a single block.

QUICK TIP

Dumb terminals – by which we really mean semi-dumb VDUs, like the VT100, because they provided more functionality than teletypes – weren't always quite so dumb. The VT103 was a VT100 to which an LSI-11 board computer board could be added to provide local intelligence. We must assume that this early personal computer wouldn't have been cheap.

X3.64 standard, on which the VT100 and its successors were based.

To use ANSI X3.64 commands for real-world applications, you're going to be including escape sequences in the textual strings you print to the terminal. However, to get a feel for them, you can echo the commands to the terminal using the `echo` command. To do that, you need to know that escape is ASCII character 33 octal, 27 decimal or 1B hexadecimal. As an example, we'll consider the escape sequence to cause text to appear with a red foreground, which is `ESC [ 31 m` (spaces only added for clarity again). So, to get a feel for using an escape sequence at the prompt, define the variable **RED** using the command `RED='\033[31m'` and then issue the command `echo -e $RED`. The next prompt is coloured in red, as is all subsequent text until you issue another escape sequence to select a different colour. And if you want an easy way of seeing the colour combinations that can be displayed this way, you can download and run the **colors.sh** script from **https://gist.github.com/sdeaton2/8450564**.



This dedicated **4010** emulator enables you to get the true experience of this classic **VDU**, even down to the bright spot as vectors were drawn.

You should also note that the colours probably won't appear exactly as you might expect from their names, and how they'd have appeared on colour terminals of old. Exact details depend on what console software you're using, and result from attempts to ensure that text is clearly legible, even on a white background. For example, the escape sequence `ESC [ 33 m` is defined as setting the foreground colour to yellow. However, because this isn't very visible against a white background, olive, mustard, pale brown or orange is often used instead. Of course, colours are really only the icing on the cake and some of the other escape sequences are, arguably, more fundamental. As a couple of examples, how about trying `ESC [ 2 J` and `ESC [ H` which clear the screen and move the cursor to the upper-left corner respectively. From here you could work up to drawing on-screen forms, but at this point you'd need to migrate to writing a script.

The Tektronix 4010, although basic, did for graphics terminals what the DEC VT100 did for their text-based alternatives.

An important benefit of using terminal control characters and escape sequences is that you issue the same commands to change colour or reposition the cursor, irrespective of what language you use. This is because these commands are interpreted and executed by the terminal, in a client-server fashion, albeit one with a very slim client. About the only difference, therefore, is in the syntax of the print statement, including how to print that all-important escape character. Needless to say, this would help in the migration of code from one language to another.

The DEC range of video terminals didn't end with the VT100 and its text-based successors, though. Hot on its heels came the VT105, which provided so-called waveform graphics, although this only allowed line graphs, histograms, strip charts and the like to be plotted. In 1981, though, the VT125 brought a full bit-mapped graphics facility to the DEC terminal range, albeit with a resolution of just 768x240. And although this was, like the VT100, confined to monochrome, colour terminals offering text plus graphics at higher resolutions followed. Graphical information was provided to the VT125 in DEC's proprietary language ReGIS (Remote Graphics Instruction Set). The commands didn't look too dissimilar to an abbreviated version of the graphics library calls we might encounter today. So, for example, without getting into the details, `V(B),[+100,+0],[+0,-10],[-100,+0],(E)` draws four vectors to form a rectangle. If you want to delve into ReGIS programming, though, *xterm* has a VT340 emulation mode. However, the default *xterm* build doesn't include ReGIS and, even when it is included, you need to start *xterm* with the **-ti vt340** flag. Arguably, though, DEC came too late to the graphics terminal market, launching its VT125 after other manufacturers had come to the fray. The most prominent such product became an industry standard for graphics terminals, and that's our next topic.

A graphics terminal – Tektronix 4010

We tend to view anything graphical as more advanced than a purely textual counterpart. So, it might be surprising to learn that our featured graphics terminal was launched five years before the VT100. The VDU in question is the 4010, produced by Tektronix, which had traditionally manufactured electronics test equipment, most notably oscilloscopes. Interestingly, Tektronix still serves this traditional market, even though its display terminal product line is long gone. That five-year advantage over DEC resulted in the 4010 and its successors being hugely influential in the world of graphics terminals. Indeed, several other manufacturers of terminals went on to emulate the 4010. In fact, its syntax was an alternative to ReGIS in

CREDIT: Roland Huisman

DEC's VT340. However, the 4010 was very different from the VT340 in a one fundamental respect.

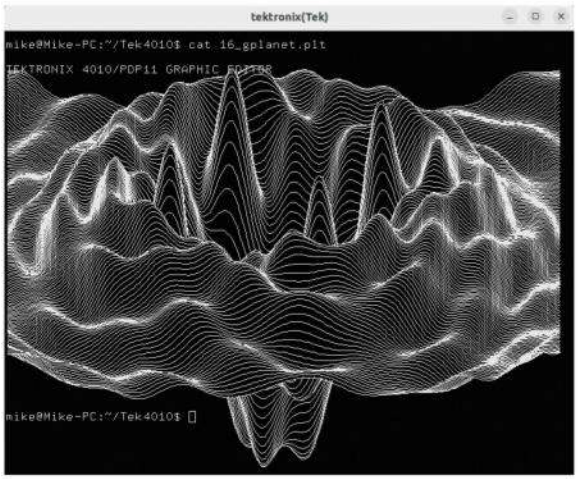
Today, we take it for granted that a computer graphics display is built around semiconductor memory that stores the colour of every pixel on the screen. Not so with the Tektronix 4010, though, as this terminal didn't have pixels. With memory being so expensive in the early '70s, the 4010 employed a storage screen. To cut a long story short, images were written to screen as vectors by scanning an electron beam across the surface of the screen from the start to the end point. The display screen's hardware caused anything written to it – which appeared as a green image on a darker green background – to remain on screen without having to be refreshed. Because graphical information was written as vectors rather than pixels, the resolution could be thought of as infinite, but the commands that wrote to screen restricted the addressable resolution to 1,024x1,024, of which only 1,024x780 could be displayed on the 3:4 aspect ratio screen. The storage capability of the screen meant it wasn't possible to erase and overwrite just a part of the screen. Instead, the screen could only be erased in its entirety.

You'll recall that we referred to the use of VDUs with minicomputers as representing the first step in the evolution of the client-server model. However, the Tektronix 4010 didn't incorporate any local intelligence, in the normal sense of the word. This is not surprising, since the first microprocessor, the Intel 4004, had been released only a year or so before the 4010. What's more, as a 4-bit design, running at 750kHz, the 4004 probably wouldn't have had the necessary power. That isn't to say the 4004 didn't have a degree of processing capability, but that was all provided by custom-designed electronics to handle the communication to and from the computer, interpret the incoming data stream, and drive the display screen.

As with the VT100, there's a good chance that you already have some software on your PC that can interpret Tektronix 4010 commands. However, that probably won't be your default console app, so you'll have to use *xterm* instead, starting it with the **-t** flag to select Tektronix 4010 emulation. And to get a feel for graphics 4040-style, we suggest you print some files of 4010 data streams to *xterm* using *cat*. See later, when we discuss a dedicated 4010 emulator, for a source of some suitable files. In our experience, *xterm* displays most 4010 data files correctly, and it does so quickly. However, it doesn't totally allow you to relive the 4010 experience, even if you choose green as a foreground colour instead of the default white. This is partly because it's too fast, and partially because it doesn't attempt to emulate the 4010's storage screen. *Xterm* might be exactly what you need it you want to run software that was written to output to a Tektronix 4010 or one of its early successors. But if you want the true retro experience, there's a better solution.

That solution is to use the dedicated emulator of the Tektronix 4010 at **https://github.com/rricharz/Tek4010**. The software also emulates the 4013, 4014 and 4015, which were similar but had larger screens and/or additional instructions. Installation instructions are provided and, in our experience, the process is trouble-free. There are also lots of so-called plot files, which have a .plt extension, these being sample files of the data stream as sent from a minicomputer to a 4010. You'll refer to one of these files in the command to run the emulator, for example `tek4010 -noexit cat dodecagon.plt`. The first thing you'll notice when you run the emulator is that it's slow – very much slower than *xterm*, which renders files almost instantaneously – so it gives a much better idea of what it was like to use a 4010. What's more, as the image builds up on screen, you'll notice that the vector currently being written is in a very light shade of green, which reduces in intensity to a dark green shortly afterwards. Again, this emulates the true Tektronix 4010 experience, and was a function of the working of the storage screen.

To genuinely get a feel for the 4010 from the programmer's perspective, though, you might want to try hacking some code. If so, we investigate the way in which data had to be presented to the terminal in the first boxout *(page 69)*. Welcome to the '70s. LXF



Unlike most **terminal** applications, **xterm** does a pretty good job of **interpreting** and displaying **Tektronix 4010** commands.

TEKTRONIX 4010 INSTRUCTIONS

If you're intent on writing graphics software for the Tektronix 4010 you'll need to consult the user manual. However, to whet your appetite, here's the gist of how vectors were written to screen.

The 4010 had both an alpha mode, in which ASCII characters were printed as text, and a graphics plot mode in which they defined start and end points of vectors. So, to create a graphical masterpiece, the first thing you need to do is send the ASCII `GS` character, which has the value x1D, to select graphics plot mode. Once in this mode, X and Y value pairs for the desired on-screen positions are sent, using two ASCII characters in the range x20 to x7F for X and another two for Y. This first set of coordinates defines the start point of a vector, and all subsequent sets of coordinates define the end point and the start point of the next vector. In other words, multiple sets of coordinates after a `GS` caused a line of multiple segments to be drawn. When that line is finished, if another multi-sector line is required, another `GS` character must be issued. At the end of the plot, the ASCII `US` character (x1F) or `CR` (x0D) is used to select alpha mode. And apart from the exact details of calculating the ASCII character pairs for each X and Y position (see the manual), that's all you need to know.

QUICK TIP

You might want to consult user manuals for our two featured terminals. The relevant links are https://bit.ly/lxf317vt100 and https://bit.ly/lxf3174010. Most Linux consoles/terminal apps support more ANSI X3.64 commands than the VT100 – for colour, for example – but you'll have to search out the information for your chosen software.

GIMP

Credit: www.gimp.org

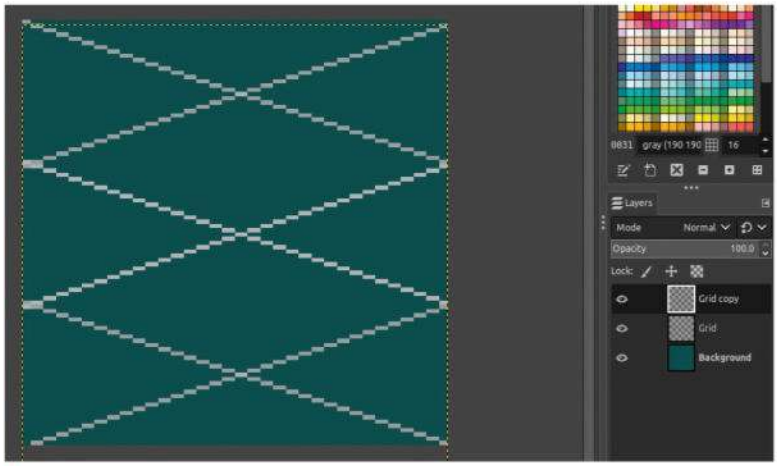
Create old-school pixel art images

Still stuck with the ancient GIMP 2, **Neil Mohr** decides to go even more retro and explore how you can create classic pixel art with it.

OUR EXPERT

Neil Mohr grew up playing Ultimate Play The Game releases on the Spectrum, so knows all about chunky pixels.

Think of retro games and you'll probably think of the pixellated look of titles released on consoles such as the NES. Games such as *Bomberman* and *Kid Icarus* worked within the limitations of '80s technology, yet offered hours of fun and iconic looks. The style faded into obscurity with the development of more powerful game consoles and 3D graphics, but has seen a recent renaissance in indie games such as *Pepper Grinder* (**www.peppergrindergame.com**). The art is still drawn pixel by pixel using a limited palette, but is much smoother thanks to careful shading and muted hues not available to '80s game developers. One of the most popular uses of pixel art is in role-playing games based on an isometric grid. We're going to create an example character (or sprite) for use in such a game.



You can quickly create a grid using layers or copying and pasting lines and then flipping.

If you fancy creating your own sprite or just want to use it as an excuse to fire up *GIMP* and learn how to use some of its tools, creating pixel art can be both fun and educational, and is a great way to pick up some *GIMP* skills. What's interesting about pixel art is that to do it well, you need to go into your *GIMP* settings and adjust the canvas size, create a drawing grid for guidelines, tweak the drawing tools and ideally adjust and create a set palette. Before we actually create a character from scratch, we'll first look at all that, plus some basics on shading, shadows and dithering, so you have some useful techniques under your belt.

QUICK TIP

Make full use of the layer system and don't overlook the Group option, so you can, erm, group layers together.

Prepare your tools

Start by creating a new image in *GIMP*. A canvas size of 100x100 pixels (perhaps 200x200 at a push) is plenty – much larger pieces of pixel art are possible, and look very impressive, but take weeks of work to complete.

Now select the Pencil tool. This, Fill and Zoom are the only tools we'll be using, but first we need to alter some settings, so we can draw individual pixels. With the pencil selected, click Brush Type and select 1 Pixel – change the size to 1 and turn dynamics off. Deselect Apply Jitter and Smooth Stroke. You can save it as a preset by clicking the blue floppy disk icon at the bottom-left of the toolbox. Click on New Tool Preset, give it a handy name like Pixel Art, then click the floppy disk icon to save it.

Prepare the canvas

Use the Zoom tool to take a closer look at your image (about 550% should be fine, although it depends on your monitor's resolution). You want to be able to see and manipulate each pixel easily. Make sure you zoom out periodically while you're working.

We need to make careful use of shadows and highlights to ensure our pixel art character looks three-dimensional. Select the Bucket Fill tool and make the whole image a medium grey colour. This neutral background shade makes it easier to judge how bright to make highlights, and how dark to make shadows.

Now we'll create an isometric grid as a guide. Create a new layer and call it Grid. Select the pencil tool and choose a different shade of grey. Click the top-left pixel of your image, hold Shift and move your mouse pointer to the right-hand side. When the coordinates at the bottom-left read 99 33, click to draw a line.

Click Ctrl+C to copy the line and Ctrl+V to paste it, then select the Move tool and move the pasted line a

little below the original. Click on the background to deselect the line, then copy and paste both lines. Repeat the process until the image is filled with evenly spaced diagonal lines.

Now copy and paste all the diagonal lines, but don't deselect them. Instead, click Tools > Transform Tools > Flip. Click once anywhere on your image to flip the copied lines. This gives you an isometric grid. Because it's on its own layer, you can delete it easily when you've finished.

Check out https://opengameart.org for inspiration, such as this countryside pixel art.

Playing with palettes

We mention the use of colour palettes in the export boxout *(over the page)*. Hopefully it's obvious that if you're designing a pixel art world, you'll want to use the same selection of bright, medium and dark shades of colours throughout. Think of a bush – it'll have at least light and dark leaves, with likely other shades in between. As modern computers have billions of colours, you'll define the small number you want in a palette and make your selection from there as you draw.

We're not going to get into palette creation but we do need to look at the *GIMP* Palettes dock and then the Colours dock. Let's select a pre-created *GIMP* palette – select Windows > Dockable Dialogs > Palettes. We suggest selecting Named Colors for now; this is a decent selection of colour shades we can get by with, but if you want to load and save palettes, this is where to do it, via the tiny top-right dock menu.

One thing to note before we continue is that if you want to edit or change the existing palette's colours – which is frankly understandable – you can't use a built-in palette as they're read-only. This isn't an issue: right-click on the Named Colors palette and select Duplicate Palette. This creates a user copy of the palette that we can edit. This is easy, too: double-click on the palette preview icon, then double-click on the colour you'd like to edit. This opens a standard colour picker, so you can adjust shades and hues as you like, but be aware that you can't undo changes made here.

Once the palette is selected, nothing as such happens – the question being, how can you select colours from the palette to use? To make use of the palette, select Windows > Dockable Dialogs > Colours. You'll see the normal *GIMP* full-colour selector. Spot the buttons that run along the top and click Palette (it looks like a paint palette), and you're good to go.

Sketching it out

To get going, many characters can be roughed out with basic geometric shapes: the body and head as blobs, arms as cylinders, feet as rounded wedges, and so on. You can sketch these out quickly in black, then fill them in, add shadowing and finally create highlights. We'll just start with drawing an egg as a basic example.

If you want to keep things simple, you can just draw a standard circle. *GIMP* does provide a Symmetry

» IT'S ALL PROJECTION

We're just introducing the basics of pixel art drawing and how you can retool *GIMP* to get you started. If you were seriously interested in creating pixel art for a specific purpose, usually within a game, you'd need to take a more serious look at projection, never mind potential palettes.

Orthographic projection is an engineering term for looking at a 3D object or scene 'projected' on to a 2D surface. In engineering drawings, you'd typically have front, side and top projections of an object. Side-view projections are how many side-scrolling shooters and adventures are displayed, while top-down projections are great for vehicle or tactical-style games. There's even a hybrid of the two if you imagine looking at a building head-on, lifting the camera up and angled down by 45 degrees, so you can't see the sides of the house but you can see the front and part of the roof, often used in cute RPGs.

We're using the more complex isometric projection, which enables a 3D feel by angling the camera to around 35 degrees (true isometric is 30 degrees) on the X-Z axis and then angling down 30 degrees. There is an oblique projection, which is used far less as it tends to look awkward, though top-down oblique has some successes.

The 35-degree angle is used to create smoother base lines; if you used a true 30-degree angle, you'd get jaggies on a low-resolution display. If you were creating a game world, you'd decide on a base tile size: 32x16, 64x32 or 128x64 – because of the perspective, tiles are twice as wide as they are deep. If you're interested in more details about gaming graphic design, see **https://bit.ly/lxf317iso** for an introduction. For more on the specifics, try **https://bit.ly/lxf317project**.

QUICK TIP

Don't forget **that you can** easily duplicate and merge layers, making **it very easy to** replicate parts.

CREDIT: Whispers of Avalon.

Our little test run, creating a pixel art egg with shadow and shading.

Painting tool, and this mirrors anything you draw, which is ideal for sketching out bodies that you want to be symmetrical, oddly enough. Switch this on via Window > Dockable Dialogs > Symmetry Painting and Symmetry Painting > Symmetry > Mirror, and select Vertical. A vertical guideline appears – just remember to deselect Vertical once you're done.

Before starting, it's worth thinking about layers, because the better use of layers you make, the easier your life is. We suggest choosing a neutral background colour, then creating a second layer and sketching out your blob on that in black with the pencil. To be honest, at this stage it doesn't really matter, because we're just experimenting, so sketch out a blob and then fill it with a solid, ideally light, colour, so we can add darker shades for shadowing. We'll just use white to keep things simple.

Selections and shadows

Use the selection tool, ensuring Feather is off in its Dock options, and select inside the shape. If you create a new blank layer when you draw, this selection will restrict drawing to within your shape's outline shape. What you want to do now is draw a graduated shadow around the bottom shape of your egg. Do a large area in light grey, a smaller area below this in dark grey, and then around the bottom edge in your darkest shade of grey. You could do each of these on their own layer if you wanted.

For the final touch, create a layer below the main egg and draw a shadow for the object – this tends to ground any object and make it feel as though it's on the floor rather than floating above it. You can also try to create an umbra and penumbra, with two layers of lighter and darker shadows. For this you might also want to experiment with dithering. Once you've created your shadow blob under the egg, select the Colours > Dither menu. One last point: if you switch to the Erase tool, ensure you select the Hard Edge option under Dynamic Options, because this ignores any softness in the selected brush to preserve your pixel art style.

At this point, hopefully you've learnt the basic tools and techniques to start experimenting with creating some pixel art. To be honest, *GIMP* isn't the best tool for the job; there are dedicated pixel art applications – check out *Pixelorama* for one (**https://github.com/Orama-Interactive/Pixelorama**) or there's *LibreSprite* (**https://libresprite.github.io**). LXF

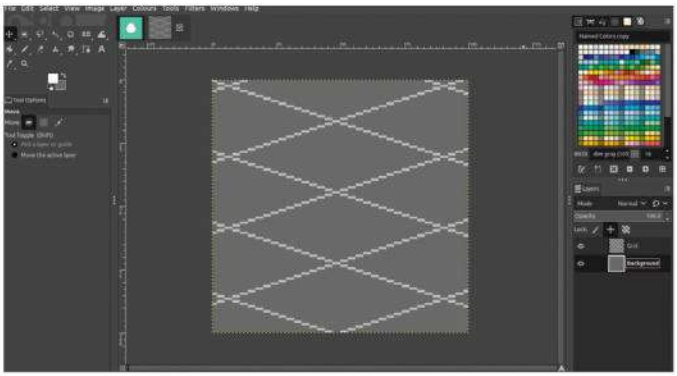
» PREPARE TO EXPORT

Pixel art has its own special requirements when it comes to file formats for saving. The key rule is that JPEG is out because it uses a lossy compression that is totally unsuitable for pixel art. If we were getting serious about creating a range of images, you'd want to put your mind to the colour palette, too. While modern PCs enable 32-bit colour (billions of colours), for pixel art you'll only want to use a handful of carefully selected colours in a swatch that you can save and reuse across all your related images. If you're interested, there are websites dedicated to just pixel art palettes, such as **https://lospec.com/palette-list**.

Once you're happy, click on the Grid and Background layers in the right-hand palette and delete them. Now click File > Export As. We want to keep our transparent background, which means saving it in either PNG or GIF format. Don't save it as JPEG – this format doesn't support transparency and, as mentioned, the compression will ruin your intricate art.

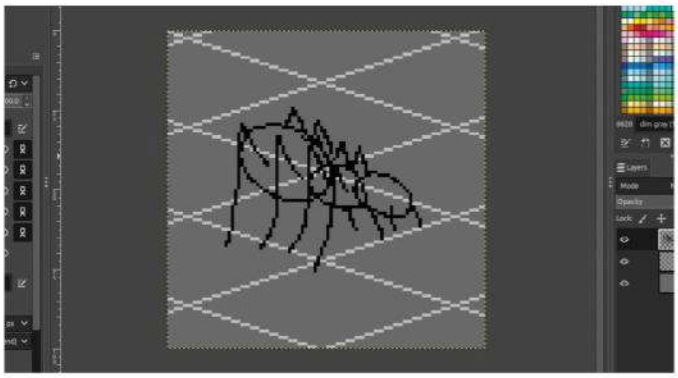
Select your preferred filetype from the drop-down list and give your artwork a name. In the next dialog box, uncheck Save Background Colour and Save Colour Values From Transparent Pixels, and then click Export. Your finished pixel art is now saved with a transparent background, ready for use in a game, as an icon on your desktop, or on a web page.

DRAWING PIXEL ART



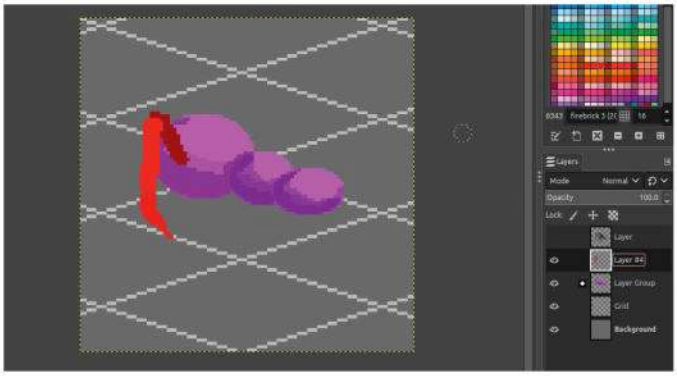
1 Pick a subject

Create a new layer and call it Pixel Art. The next step is largely up to you, but we're going to draw a spider-like creature, which could be an enemy in a computer game. Make sure you use the grid as a guide to help you get the right perspective, and don't use any tools other than your customised Pencil and Fill.



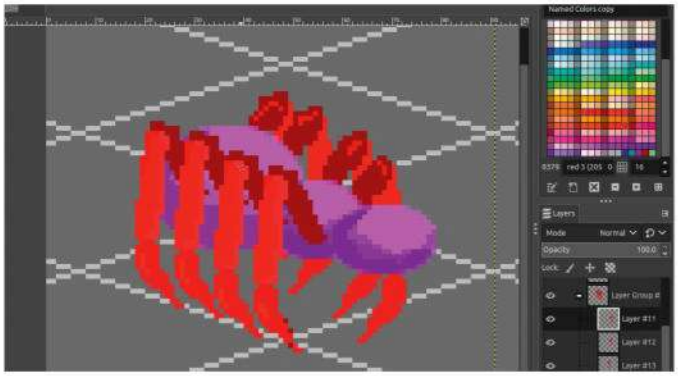
2 Sketch it out

Take a leaf out of the classic artist sketchbook and start simple. Rather than trying to get every pixel perfect first time, make a rough sketch. It's very easy to edit and refine as you go along. Just make sure you use the isometric grid as a guide, and bear in mind that your character is a three-dimensional object.



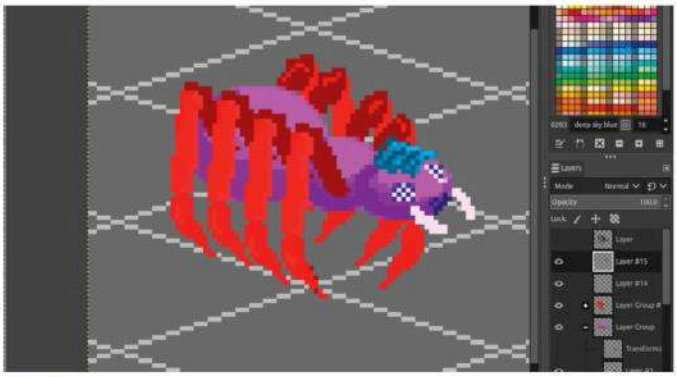
3 Go geometric

Start by drawing basic shapes on separate layers to fill out body parts. Use your shading knowledge to add a 3D feel to these and layer them in the order they're seen. Use dark edges to make sections stand out. If you have a drawing tablet, you could draw some of the body parts, such as the legs, freehand.



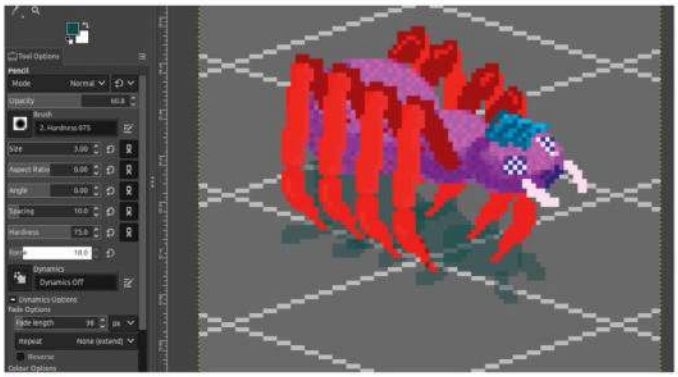
4 Refine it

Once you've got a rough sketch, try filling each section with a different colour to make them easier to identify. You can then decide where your light source is and begin adding shade and highlights accordingly. Feel free to tweak; here we've decided to make our spider's legs tilt inward to make it look more alien.



5 Add texture and details

Try to create the impression of different textures rather than making your character uniformly smooth. Here, we've decided to give our spider some fur on its head, using 'strokes' of differing shades to represent sections of hair. Zoom out regularly to check that your texture effects work when viewed at 100%.



6 Try dithering

Early games required pixel art that used as few colours as possible. One way to get around such limitations was dithering, a chequerboard of lighter and darker pixels that blend together to look like a single, mid-toned colour. Give it a try! You can create an interesting texture if the two colours are quite different.

Use some admin-fu to master snapshots

Most users don't much care about the underlying disk management system LVM, but **Stuart Burns** reveals one of its most useful features.

Among Logical Volume Manager's (LVM) many features, one of the best is snapshots, which is particularly useful for system administrators. Snapshots allow for the creation of a point-in-time copy of a logical volume, which can be useful for upgrade, backup and recovery purposes.

This article explores the utility of LVM snapshots and provides practical scenarios where they can be applied effectively. We do assume some familiarity with basic LVM and its setup, so see the boxout *(below)* for more getting-started advice.

Most administrators use snapshots at the machine level, but sometimes machine-level snapshots aren't possible – for example, physical machines and virtual machines from providers who don't provide a snapshot service. Logical Volume Manager (LVM) can provide flexible snapshots in Linux no matter its underlying configuration, as long as it has LVM.

LVM snapshots are not a backup (in this scenario) but they allow the administrator to create a point-in-time copy of a logical disk before making any major changes, such as upgrading an application.

Should the application upgrade go wrong, you can roll back the snapshot to its creation time, reboot and all is well. Sounds awesome, right? Before jumping in, however, there are a few notes of caution…

1 Most Linux vendors suggest not using this before a major OS upgrade – it gets too messy, because there are so many moving parts.

2 The snapshots have to be appropriately sized to cope with all the disk writes that occur for the lifespan of the snapshot. If the snapshot runs out of space, it will be corrupted and unusable.

3 LVM snapshots deal with volumes, not entire disks or servers. You need to understand the application you are upgrading and where it lives.

4 To take a snapshot, there needs to be sufficient disk space on the volume group in question. By default, Ubuntu uses all the disk space available.

Rather than risk your working system, use a VM. We created a basic VM and installed Ubuntu with the defaults, then added a second 10GB disk and created volume group **data-vg** and a logical volume on top, **new_log_vol**, sized to 2GB, leaving 8GB of the volume group free. We have the logical disk mounted at **/data** with an ext4 filesystem.

Understanding snapshots

An LVM snapshot is a read-only or read-write copy of a logical volume at a specific point in time. The snapshot volume initially occupies very little space, only increasing in size as changes are made to the original volume but diverted to the snapshot (copy on write, to use the technical term). This makes snapshots a highly efficient way to manage backups and system states without duplicating the entire volume's data.

One of the most common uses of LVM snapshots is for backup purposes. When taking a backup,

DISCOVER MORE ABOUT LVM

If you're new to LVM, it's time to do some reading up. For budding sysadmins, Red Hat offers an introduction at **www.redhat.com/sysadmin/lvm-vs-partitioning**, while we ran features on LVM storage in **LXF205** and **LXF252**.

The basic outline is that LVM presents storage as logical volumes, which look just like regular partitions, but can span multiple physical partitions or even drives – they become things you don't need to worry about. Regular partitions, drives or other block devices (physical volumes, or PVs) can be grouped into volume groups (VGs, the LVM abstraction of a physical disk), then logical volumes (LVs) can be created on top.

LVs are then totally free of physical boundaries and, unlike old MS-DOS partitioning – where you are limited to three primary partitions, and if you need more, you have to create logical partitions within those – you can have as many of them as you want.

Using LVM, we can easily add more space to a filesystem, even if that space is on another drive. Once we've added the physical volume representing the new drive or partition to the relevant volume group, there's no notion of adjacent space – it's all fair game. You can create a filesystem (anything from FAT to ZFS) on an LV exactly as you would on a regular partition, but first you need to set up your VG, and the first step is to create the PVs on the drives that will house it.

consistency is key, especially for databases and filesystems in active use. By creating a snapshot, you can capture the state of the data at a particular moment without shutting down the service. Before starting any LVM operations, see how the volume group layout looks using the command:

```
$ sudo vgs
[sudo] password for sysadmin:
VG        #PV #LV #SN Attr   VSize   VFree
data-vg     1   1   0 wz--n- <10.00g <8.00g
ubuntu-vg   1   1   0 wz--n- <60.95g 30.47g
```

Using the below example, create a 1GB snapshot named **lv_data_snap** for the **data** folder. In reality, the snapshot should be a lot bigger. As warned earlier, a full snapshot is a useless snapshot.

```
$ lvcreate --size 1G --snapshot --name lv_data_snap /dev/data-vg/new_log_vol
```

All being well, it should finish with the message `Logical volume lv_data_snap created.` The administrator can now use the `lvs` command to see if there is now a new entry for our snapshot:

```
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert
lv_data_snap data-vg swi-a-s--- 1.00g new_log_vol 0.01
```

If you now create an example text file, and put an image or two in the **/data** folder, then use the `lvs` command again, you will see the **Data**% has increased. This is the percentage of snapshot space used.

To remove the snapshot (if the upgrade went well), it is a case of simply using the command below:

```
$ lvremove data-vg/new_log_vol
```

You'll see an error saying that merge will occur on next activation. To force an activation, it's easiest to just reboot. Upon reboot and logging in, doing an `lvs` shows that the snapshot has gone and the files that were present earlier are now permanently available.

If the upgrade goes wrong, you can revert to the point-in-time copy (think of it as a rollback) with:

```
$ lvconvert --merge vg00/lvol1_snap
```

All this is great but LVM has an even better superpower: if you create a snapshot, it is a frozen-in-time copy. With this 'no change', it makes it ideal as a backup medium and helps ensure files are backed up in a consistent state. Of course, it can be automated, but as a quick and simple one-off work-through, this consists of mounting the snapshot so the data can be copied. For example:

```
$ sudo mkdir /mnt/backup
$ sudo mount /dev/data-vg/new_log_vol /mnt/backup
$ sudo rsync -azv --progress -e ssh /mnt/backup/ stu@mybackupsever:/srv/data/home/
```

Snap to it

LVM snapshots are a versatile and efficient tool for sysadmins, offering significant benefits in backup, testing, system upgrades and disaster recovery. By incorporating snapshots into your workflow, you can enhance the reliability and resilience of your systems, ensuring quick recovery from errors and efficient management of data states. Embracing LVM snapshots not only simplifies your administrative tasks but also provides peace of mind knowing you have robust mechanisms in place to safeguard your data. LXF

```
sysadmin@lvmtest:~$ sudo -i
[sudo] password for sysadmin:
root@lvmtest:~# lvs
  LV          VG        Attr       LSize  Pool Origin      Data%  Meta%  Move Log Cpy%Sync Convert
  mysnap      data-vg   swi-a-s---  1.00g      new_log_vol 0.73
  new_log_vol data-vg   owi-aos---  2.00g
  ubuntu-lv   ubuntu-vg -wi-ao---- 30.47g
root@lvmtest:~#
```

The lvs command happily lists your logical volumes, including snapshots.

```
  --- Logical volume ---
  LV Path                /dev/data-vg/mysnap
  LV Name                mysnap
  VG Name                data-vg
  LV UUID                7u42eO-0bsZ-oHAJ-3mqH-f1MR-f12L-ldirqj
  LV Write Access        read/write
  LV Creation host, time lvmtest, 2024-06-02 20:47:26 +0000
  LV snapshot status     active destination for new_log_vol
  LV Status              available
  # open                 0
  LV Size                2.00 GiB
  Current LE             512
  COW-table size         1.00 GiB
  COW-table LE           256
  Allocated to snapshot  0.73%
  Snapshot chunk size    4.00 KiB
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:4
```

To keep an eye on your snapshot, use the lvdisplay to get details on what it's up to.

The Raspberry Pi Foundation has confirmed its plans for an initial public offering.

Stuart Burns is a Linux administrator for a Fortune 500 company specialising in Linux.

» SLICES OF PI TO GO ON SALE

The world of Linux has been relatively quiet. Some of the news includes open source hardware maker System76 releasing a slew of new and improved laptops and desktops at more appealing price points.

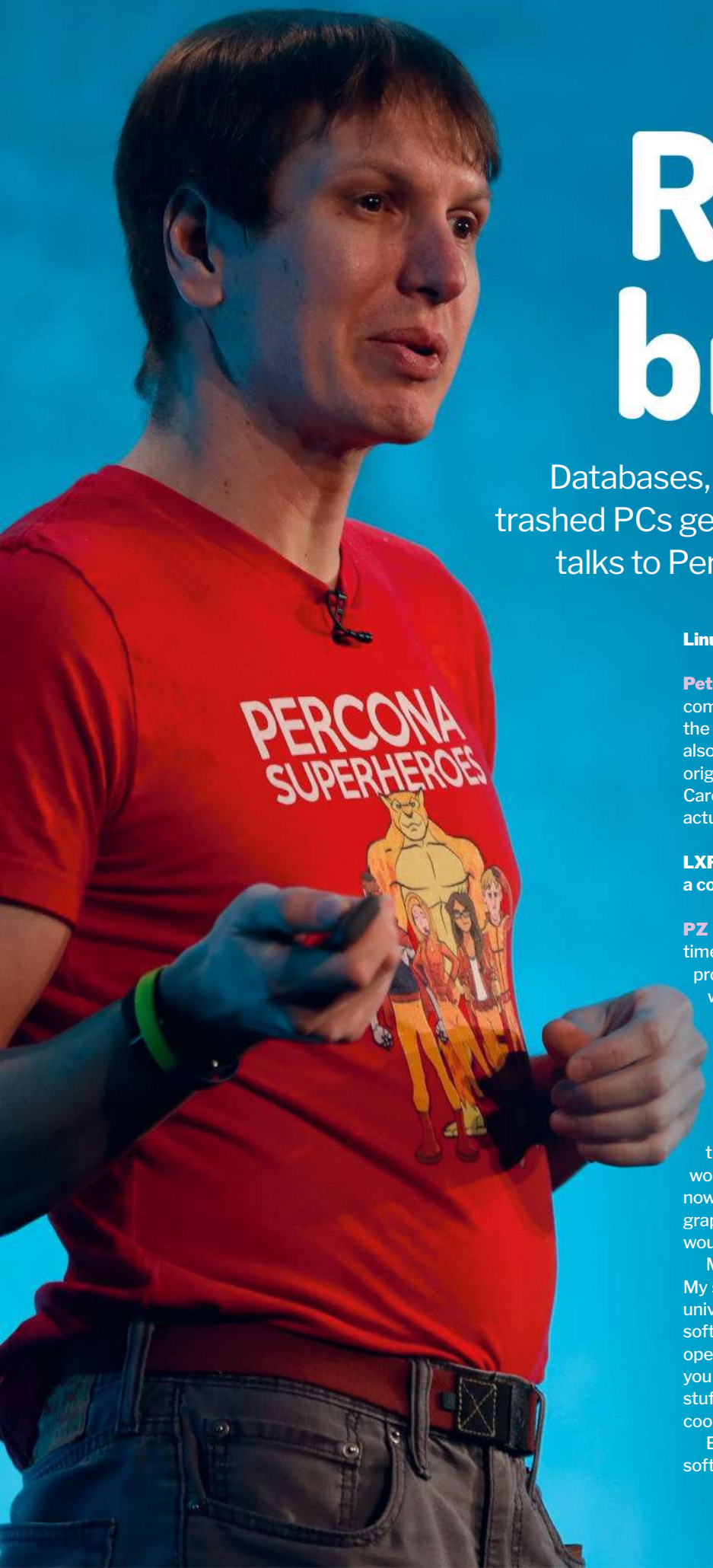
Some of the highlights include better screen displays and GPUs (Intel ARC features strongly). What is interesting is that it is not just one laptop that has been re-envisioned but several. This means the consumer now has a better choice: big and powerful (and relatively heavy) or small with a long battery life. The choice, as they say, is yours. More information can be found at **https://bit.ly/3V9pMDr**.

Also of note is that Raspberry Pi is moving closer to listing on the UK stock exchange. It was first discussed just before Covid, but for obvious reasons, was delayed. Fast-forward to today, and who would have thought back in 2013 this could ever happen? How far we have come!

Some believe this could be detrimental to the whole ecosystem. I believe that's not the case. Raspberry Pi is a rare UK success story. The bulk of Pis are even made in Wales.

There are endless different SBC (single-board computer) systems available. None of them has what Raspberry Pi has: excellent documentation, the goodwill of hobbyists and a lot of companies adopting it into their stack, as it is stable and predictable. You can read the notification of the intention to float at **https://bit.ly/4bH7joN**.

Only time will tell, but the Raspberry Pi Foundation has promised that this won't distract from its education arm or the development of new systems.



Root & branch

Databases, observability, open core and trashed PCs get a mention as **Linux Format** talks to Percona founder **Peter Zaitsev**.

Linux Format Tell us a little about yourself, Peter.

Peter Zaitsev I'm now the founder of a few companies in the open source space, Percona being the largest and the most well known of those. I'm also very active in the open source ecosystem. I'm originally from Russia and now live in the US, North Carolina. I also lived for a bit in London and that's actually where Percona got started back in 2006.

LXF What are your first memories of using a computer?

PZ I got my first computer from a trash can. At the time I was staying for a year in Sweden, my dad was a professor at a university. I think some local company was refreshing their computers, and throwing away perfectly good IBMs. So it had a monochrome display and half a MB of RAM, which I got to use.

That was a fantastic experience, because you don't have the latest and greatest stuff. It's kind of slow and you don't have the resources – you have to figure things out, how to make things work that are not supposed to work, and I think that was wonderful. Comparing that to my own computer right now, you have this wonderful PC with a powerful graphics card. You can run all these fancy games. Why would you want to learn assembly language?

My involvement with open source is interesting. My start was back in Russia when I was 19. I was in university at that time and came to the choice of software to use. It was an interesting time, because open source software was not what it is right now. If you think about the late '90s, Linux and all that kind of stuff was just starting. It was kind of a joke. It was very cool but not as mature or powerful as it is today.

Especially being in Russia, all software was free software! Nobody would think about such things as

It's **not** just **performance**, **Coroot breaks down costs**, **too**.

licences – you could go to a local store down the street and buy a single CD, which would have everything you ever needed for maybe 50 cents! So, we made a deliberate choice not to use that, but instead take the high road of using Linux, *MySQL*, PHP – the LAMP stack – and, well, the rest is history.

What was also very interesting for me is that open source is a community. If you're using *Oracle* or Windows and you find a bug or you don't understand something, what are you gonna do? Maybe you call support and you get somebody asking if you've tried turning it on and off again.

But if I found some problem with the Linux kernel, I would write to the Linux mailing list and have Linus Torvalds's response! You can have other people who are senior experts give you their time and attention. That was wonderful being involved so early on.

What excites me now is being involved in the early stages of projects, because the community is there. You can have those kinds of absolutely fantastic folks who have time to talk to you. Sure, over time things change, when a project has millions of users, people at the top of that project won't have time for the average Joe, but at the start, when there's tens of users, well, there's these awesome people to talk to.

LXF We've heard that sort of narrative come up before. People enter the community maintaining software they need, like Mark Shuttleworth starting with Debian looking after Apache.

PZ Absolutely. I think an interesting thing is I'm also an advisor to a number of open source projects at an early stage. I say to them, "Hey, guys, writing the code, it's fantastic, but you really need to give time to your open source community." As you're building those first 100 and then thousands of users, you have to give them your love, locate their problems and resolve them. That's how you get them to love a project, by getting that help from the founder him or herself, and that's how you learn to understand your users better.

It's an interesting thing in open source that you often see – the most gifted engineers tend to live on a different planet, a different reality. Things that are complicated for normal people, they don't even see as a problem.

LXF They think in different ways and approach problems from a different angle from the average Joe.

PZ It's very important thing to know and understand. It's not only your average Joe, right, but it's many different Joes. That's the other thing as well: how you approach a problem. People approach problems and innovation in different ways, and understanding that is what you need for a product to be successful.

LXF So, how did you go from studying at university to making Percona a success?

PZ I was an engineer at MySQL. I would say it was one of those romantic open source companies. As engineers we're not there to make money off the stock options or the bottom line, it was about making a difference. We were much more proud in terms of how many small startups or websites *MySQL* would power.

Around the time I left, that was changing. MySQL had raised a couple of rounds of venture capital and said, "Hey, guys, the most important thing here is to provide the best returns to our shareholders," which did not motivate me as an engineer.

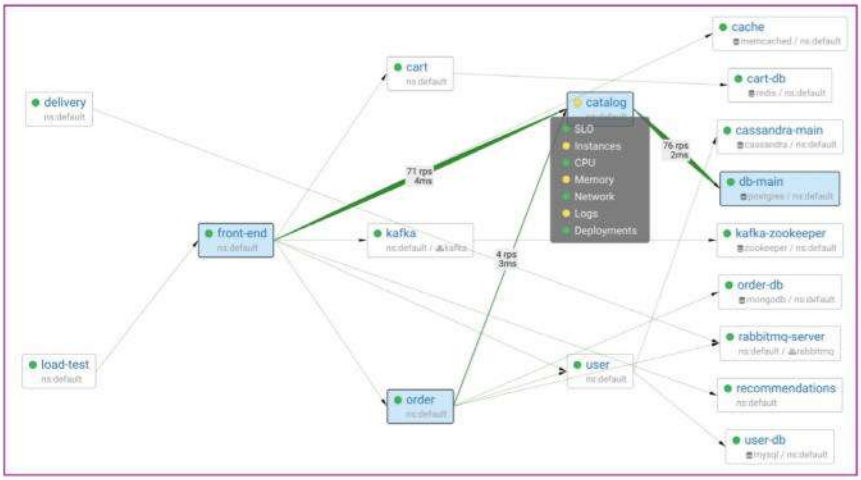
So, I decided to leave and start my own company. That's how Percona got started. What was interesting is that in the first year, many people said, "Oh, Peter is just the geeky type. He'll tire of this in a year or two, and get a proper job," and you know, it didn't happen.

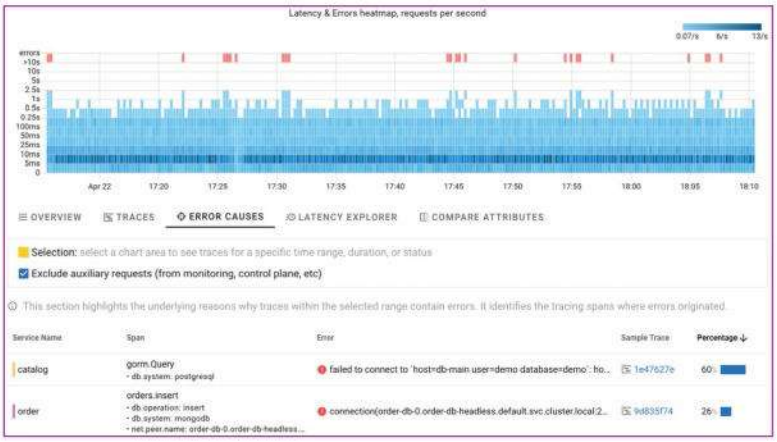
There were some factors like timing and I was lucky and got some good early customers, and I was able to build a very good team to help me. As someone who is a geek, you probably only want to do 20% of things the company needs to exist. For example, I would hate all that HR and administrative stuff, absolutely, I've always despised it and always will, but you can't run a company at any scale without taking care of that stuff.

If you look at the genesis of Percona, one thing stands out: putting the customer first. A lot of

Percona was **founded in** 2006 **by** Peter Zaitsev **and** Vadim Tk**achenko**.

Coroot can create service maps **that cover** 100% **of your** system **with no blind spots**.





Distributed tracing enables engineers to visualise the path of a request across components, helping identify bottlenecks, latency issues and general errors.

companies offer professional services. If you go to them, they're not going to say, "Hey guys, your application is simple enough, maybe *Postgres* would work well!" Typically, those professional service teams are extensions of the sales team, which you pay for, and that's something I wanted to break.

If you look to the US financial industry, there's a concept of fiduciary responsibility. To provide advice that's not to make the most money but is best for the customer. It doesn't always work that way, but at least there's some sort of industry ethics and laws in place. There's nothing like that in software professional services, and many professional services operate to first and foremost make more money for their company, and secondly for themselves, and thirdly perhaps then the customer gets something.

That is the thing we wanted to turn around and concentrate on, really focus on a customer solution that was very successful. I would tell my team many times, and one of the proudest moments is when you say to a customer, "You probably shouldn't do what you're asking us to. We're not going to take your money because that's not the right decision for you."

LXF Moving from Percona, you're launching *Coroot*.

PZ I'd been the Percona CEO for 17 years and that's a long time. Percona has grown pretty large and it's not been as fun for me any more. In a larger company, it's more management and less geeking out. The things that excite me are much closer to the technology, close to the code, the zeros and ones. On the Percona journey, I helped start a couple of companies in the open source space. One is Altinity (**https://altinity.com**), which provides open source solutions around *ClickHouse,* which is a data analytical technology. The other interesting one is *FerretDB* (**www.ferretdb.com**), built as a *MongoDB* alternative based on *Postgres,* it's kind of a *Mongo* front-end and *Postgres* back-end.

I was always very excited about observability, or monitoring, as we used to call it. I think that comes from the case where you look at a customer's database and systems in general, and you can see people operate them inefficiently or have a lot more downtime than they should, just because they don't understand how the overall system operates. That's what you can fix with good observability.

What excites me about *Coroot* is that I've worked with a lot of observability systems. In many cases, they are designed by geeks for experts. They say, "We're going to capture thousands of metrics every second and create hundreds of dashboards!" A lot of experts love this – they say, "I've spent a few hours on this case, looked at all those awesome graphs and I've figured out this very complicated problem. I'm a hero!" Bring it to a developer, who thinks of a database as just a thing, not something they love, or it's just one of 10 or 20 services they look after. They want something simpler.

Coroot has a different approach. We say we want to look at this simply, we're not going to resolve every single problem we can have. First, we'll tell you which component is having the problem – is it *Postgres, Radius* or *OpenAPI* giving you the problem? We'll always figure it out. In most cases, we'll allow you to resolve maybe 80% of the problems. For the remaining 20%, you can use a specialised tool or go to the vendor. That's what *Coroot* is focused on: simplicity rather than getting all the possible information. We concentrate on the essential, actionable insights that you can solve.

The second problem is if you look at a lot of observability deployments they're like Swiss cheese. People say there's certain aspects they can instrument in their systems and others they won't, because there's a complicated configuration. That means there are certain things about your system that you don't know about, as there's not a properly configured agent trying to monitor that service, and that's really problematic.

It's like the anecdote of the drunk man looking for his keys under a street light. When asked where he lost them, he says he doesn't know, but this is where he can see. We want to look for the keys where they've been lost, not just where the light is. This is why *Coroot* is easy to install and uses eBPF, with a zero-configuration installation. Many people install *Coroot* and discover things about their systems they didn't know!

LXF EBPF is pretty low level?

PZ EBPF is a kernel interface, so is kind of a low-level analogue. In the end, we have to use various kinds of low-level technologies to provide high-level awesomeness. It's not like with eBPF you can just click a button and get massive observability. *Coroot* uses eBPF to give what it exposes, like procfs and a bunch of other Linux interfaces to figure things out. Then there are further smart things you have to use to connect the data, which eBPF provides, using a system call writing to a socket or writing a file descriptor. You don't know what actual host that corresponds to and what the protocol is; you have to use some magic to connect the different pieces of information together, which requires a lot of know-how and trial and error.

LXF It sounds as though a key feature is cutting down data overload?

PZ *Coroot's* value proposition is: it's simple to install. You can install it everywhere required: *Kubernetes, Docker,* VMs, cloud. Some awesome next-gen tools only work with next-gen infrastructure like *Kubernetes*. Many companies have services and partner obligations that run on bare-metal VMs. *Coroot* can handle that.

After that, the question is: what's easy to use? You don't want all the information, just the essential

minimum, focused on actionable insights with root-cause analysis for incidents, rather than saying we'll give you lots and lots of data so you can geek out for hours. If you want that, you can still reuse the standard storage formats, connect to *Grafana* and build more graphs on the data.

Another thing I think is important is open source and cost. We can see in the observability space that Datadog is the new Oracle, in that it's expensive. If you want to do really heavy, advanced observability, it becomes even more expensive. What happens in many cases is people have to make a choice on observability about the amount of data they capture to troubleshoot an application because of the amount of money they can spend. With *Coroot,* you can get the observability you want rather than what you can afford.

LXF Coming from Percona must give you insight into what information people need from observability…

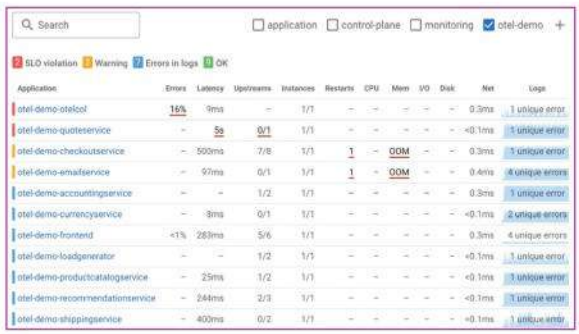
PZ What's interesting as a trend is people like to see a lot of audits. They set a lot of audits and look at what gets thrown out, and others get set on what I would call vanity metrics. So, someone can say, "My cache-hit ratio is below 95." Who cares? Looking at *Coroot,* our approach is if you want your audits to be actionable, you need to make sure there's not as much noise. You do that by focusing on service level objectives, what really matters, what impacts the user. If the user is impacted or about to be impacted, it's time to get the developer out of bed.

LXF What's your plan for making *Coroot* sustainable as a project?

PZ Looking at *Coroot,* it's going into an open core plus SaaS business. It's a conventional approach, where it's open source and gives you access to the most important stuff, with the enterprise version focused on things enterprises care about, like single sign-on and better access differentiation. People can also say that deploying and managing *Coroot* is cool, but we have a live environment with a lot of durability data, which is a task in itself, I'd prefer to have it just run in the cloud, and you can pay *Coroot* to do that for you.

LXF What are common mistakes you think open source projects make?

PZ I think a big issue at the moment is how projects change their licence. Think about HashiCorp quite recently or Elasticsearch before that – they change the licence and ditch the open source. That's not a mistake – they like to position it as a mistake because it makes them look better.



The Application Health Summary shows the status of your services.

It's much easier to build a community when it's open and then they think to monetise that, and it shifts. This has been the Silicon Valley playbook for a long time for many services. Think about Uber: we'll give you highly subsidised cheap rides until the taxis are out of business, and then we can start making money! That is just the playbook applied to open source.

That creates a loss of trust in corporate open source, especially in larger venture-funded public companies, unlike a private company. With a private founder, they can say I'm not going to maximise financial returns, I'm going to have balanced interests, and money isn't the most important thing. If you have a venture-capital-funded public company, your investors expect to make money.

Look at the open source community and how it can protect itself from that happening if there's a core open source project that's run by a foundation and commercial entities built around that. Some of the most successful open source projects are just that. Think about Linux, it's an open source non-commercial entity. If IBM does some shenanigans at Red Hat to a large extent, who cares? Linux will still survive – there's Ubuntu, there's Debian, there's a bunch of others.

Think about *Postgres*. It has a fantastic number of things, some of them open source forks and extensions, some are in the cloud, and there's a lot of commercial ones. But the core project is open source that everyone can view and make successful.

I expect people will be looking more carefully at the governance of open source projects in the future. If you want a level of certainty that one day you won't wake up to find they're changing the licence so it can make more money, if you don't want that, then an open source project with a community-controlled foundation is what you should be looking at.

LXF The relicensing issue seems to crop up time and time again. Is this just something we have to live with?

PZ Say people use Elasticsearch everywhere, promote it and contribute to it, because it's open source and they're thinking they're on a level playing field with Elastic. They're all making Elasticsearch together! Now Elastic says, "Nope, that's not the case." Well, when the next Elastic comes by, companies are not going to just say, "Great, that's open source, let's contribute our time and money," as they'll know most likely they'll have the rug pulled from under them.

There's no point for the next Elastic to do that. Just start with something like the Server Side Public License (SSPL) to begin with. I would like to see that. A lot of those kinds of moans about how open source is unfair and cloud vendors are not contributing are from people who shouldn't be starting open source projects.

Open source projects are not about making it easy for you to maximise your profits. I believe an open source project is something you want to make a difference in the world. You want to maximise access. Or it's fun for you! If you want to maximise protection of your IP rights, then guess what – there's already a proprietary business model for that. LXF

When not talking seriously about databases, Peter Zaitsev does some serious ultramarathons!

HotPicks

Drawpile » Ungoogled-chromium » Gopeed » Bitwarden » Errands » Tuba » PDFsam » Widelands » Dr Robotnik's Ring Racers » Shutter » Obfuscate

Mayank Sharma
has accumulated a lot of bad karma driving a diesel-guzzling 4WD, which he is undoing by digging up sustainable-by-design open source software.

COLLABORATIVE WHITEBOARD

Drawpile

Version: 2.2.1
Web: https://drawpile.net

Krita offers everything you need to create digital art on the Linux desktop, but what if you need to work with other artists to conjure up your masterpiece? *Drawpile* is a cross-platform drawing app whose main goal is to let multiple people work on the same whiteboard simultaneously.

Drawpile is officially available as both a Flatpak and an AppImage. It's best to use the AppImage to experience the app's networking features. Just grab the AppImage file, then make it an executable from the file manager or with `chmod +x`.

The app has everything you would expect from a freehand digital drawing app, including hundreds of brushes, the ability to work with layers, and all kinds of editing tools.

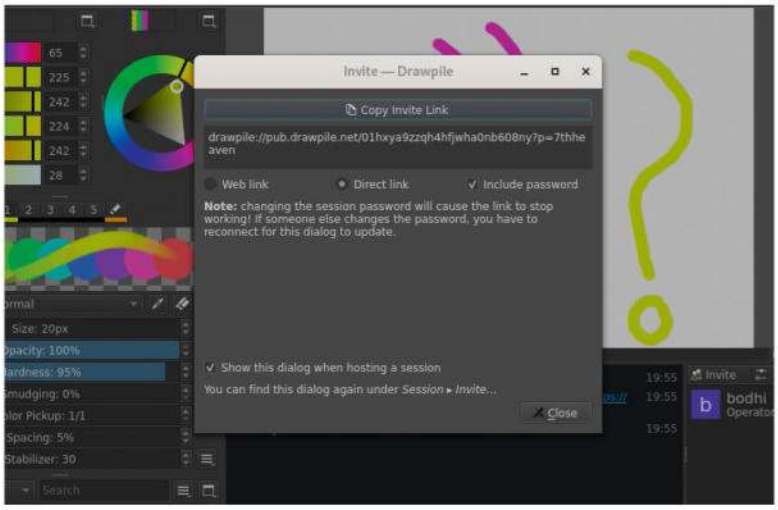
However, the best part about *Drawpile* is that it allows the canvas to be shared between several users. Inside the app, head to Sessions > Host, and give the session a title.

If you want your session to be private, you can give it a password. That way, only people to whom you give the password can join. If you don't assign a password to your session, it is public and anyone can join it.

You can either host your session on a public server, which is the easier option and costs nothing, or on your computer, for which you need to have necessary networking chops.

The public *Drawpile* server **pub.drawpile.net** is already preselected by default. All you need to do is click on Host. The app now prompts you to log in to your *Drawpile* account, or you can select the Continue Without Account option, where you only have to specify a username.

Your session is now available and others can join it. *Drawpile* gives you a link that you can use to invite friends. You can now pass the link to people with whom you want to collaborate. In their *Drawpile* instance, they have to head to Session > Join, and enter the link you've sent them.



By default, **D**rawpile includes your password in the invite **URL**, but you can uncheck the **I**nclude Password option to force your friends to manually enter the password.

LET'S EXPLORE DRAWPILE...

1 Brushes dock
This dock offers all the preset brushes. The brushes have the usual properties you'd expect, such as opacity, hardness, spacing and more.

2 Brush editor
You can use the brush editor to adjust all kinds of settings for the selected brush. Refer to ***Drawpile's*** manual to get to grips with this most important dock.

3 Drawing toolbar
You can use this toolbar to draw some common geometric shapes, and it also enables you to switch between the freehand drawing and eraser modes.

4 Layers
You can use this dock to add multiple layers to keep different parts of your painting separate from each other, until you merge them when you're done.

5 User control
All those connected can chat through this dock. They can also see who is connected to the app, and can even invite more users.

WEB BROWSER

Ungoogled-chromium

Version: 124.0.6367.118-1 **Web:** https://github.com/ungoogled-software/ungoogled-chromium

Before you can install extensions from Chrome's web store, you must install the chromium-web-store extension in Ungoogled-chromium.

Back in 2008, Google open sourced the core components behind its popular browser, in the form of the *Chromium* browser. Out of the box, *Chromium* is generally considered much better than *Chrome* in terms of privacy because it lacks Google's proprietary privacy-invading services. It also serves as the base for several third-party browsers, such as the privacy-focused *Brave*.

Yet *Chromium* isn't perfect. Because it is built by Google, it still has features that call back home. For instance, *Chromium* has Google Host Detector, Google URL Tracker and even Google "Safe" Browsing, which all depend on Google.

Ungoogled-chromium is exactly what it sounds like. Think of it as *Chromium* without all the Google dependencies. The browser is available in the default repos of many popular distros, but it's best to get the latest release from Flathub with `flatpak install flathub io.github.ungoogled_software.ungoogled_chromium`.

On first launch, the browser tells you how it differs from vanilla *Chromium*. The browser retains most of the features, such as *Chrome's* UI enhancements and support for profiles. That said, stripping away all Google-originating services also comes with its downsides. For instance, *Ungoogled-chromium* loses all *Chromecast*-related capabilities.

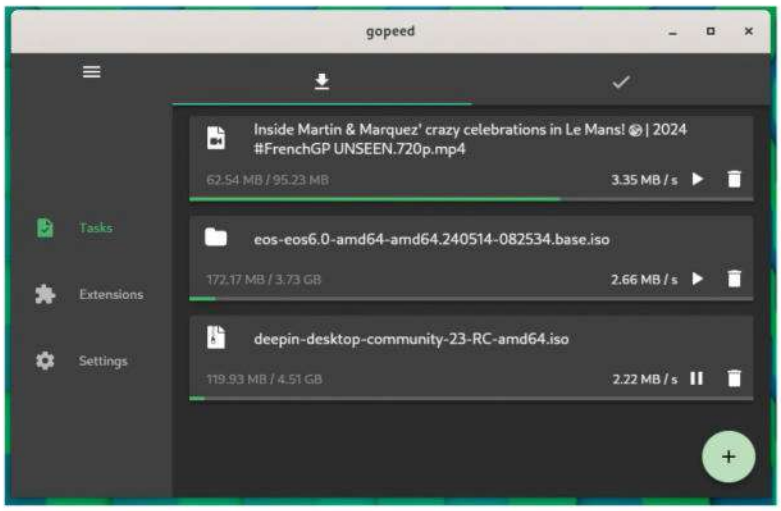
While the browser looks like regular *Chromium*, there are several notable usability differences. For starters, *Ungoogled-chromium* has no search provider out of the box. You have to head to **chrome://settings/search** to select from one of the four supported search engines. You can also continue with the No Search option, which disables searching in the omnibar.

It also disables automatic formatting of URLs in the omnibar, so it doesn't strip the http or https from a URL. It also forces pop-ups to open in new tabs instead of windows, and deletes browser history after 90 days. Virtually all of these features can be overridden.

DOWNLOAD MANAGER

Gopeed

Version: 1.5.6
Web: https://gopeed.com



To download, head to Tasks, hit the + button and enter the URL. If the URL is in the clipboard, Gopeed picks it up automatically.

Most people don't need a download manager, but if you're grabbing large files like distro ISOs, you'll enjoy the flexibility you get with a dedicated one, such as *Gopeed*.

It's available on Flathub, and you can grab it with `flatpak install flathub com.gopeed.Gopeed`. The downloader supports the three popular download protocols: HTTP, BitTorrent and Magnet. Before you get to downloading, take a moment to go over settings.

The settings pane is divided into two tabs. Use the Basic tab to customise the download destination, which by default is **~/Downloads**. The app is also set to run five concurrent downloads at most by default . Depending on your bandwidth, you can increase or decrease this number. If you're unsure, it's best to leave it as is. In the same vein, unless you know what you're doing, it's best to just review the settings for HTTP and BitTorrent protocols.

One interesting option is the pointer to the *Chrome* extension. If you are using *Chrome* or a *Chrome*-based browser such as *Brave*, click on the link to fly to the Chrome Web Store, from where you can install the extension for *Gopeed*. When installed, the browser hands all download tasks to *Gopeed* instead of the browser's built-in download manager.

At the moment, *Gopeed* only has a browser extension for *Chrome*, but there are other extensions. Switch to the Extension section and click Find Extensions. This takes you to the app's GitHub page, which lists all extensions. The most useful are those to grab videos from YouTube and X (formerly Twitter).

To install the extension, grab its GitHub URL and paste it in the Install URL field in the app, then hit the Download button next to the URL field.

PASSWORD MANAGER

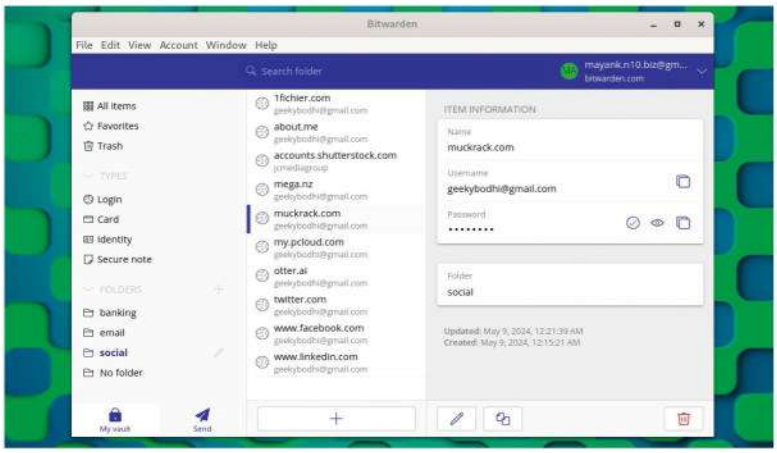
Bitwarden

Version: 2024.4.3
Web: https://bitwarden.com

We've told you many times that you should use a unique password for every account. But since not all of us are memory champions, remembering a dozen or so unique passwords isn't feasible. This is why we need password managers. While most web browsers come equipped with decent password managers, security experts believe browser-based password managers aren't as secure as dedicated ones, such as *Bitwarden*.

Bitwarden came out top in our *Roundup* of password managers (**LXF312**, *page 24*). The open source personal edition for individuals is available at no cost. You can grab it as an AppImage from its website and make it an executable from the file manager or with `chmod +x`.

When you launch it for the first time, you're asked to create a new account. This involves entering your email address and a master password. You can also create this account on *Bitwarden's* website, but the only advantage of doing so on the website is that you can choose to create the account on its EU-housed servers instead of the US-based ones.



The app has a three-pane interface. Begin by creating a folder using the + icon in the left pane. Folders are the app's mechanism for organising passwords. So, you can have a folder for banking, another for shopping, for example, and so on.

To save your credentials, click the + icon in the middle pane. This brings up the Add Item panel in the third pane. Use the pull-down menu to select the type of credential. Besides website logins, the app can protect your debit and credit cards, or your identity, or can even be used to secure notes. Select Login and enter the credentials, followed by the URL of the login page, and the folder you want to file the credentials under, and you're done. Refer to the app's documentation for more usage examples.

Bitwarden can import and export passwords in all kinds of formats, including the commonly used ones such as **CSV**, **JSON** and encrypted **JSON**.

TO-DO APP

Errands

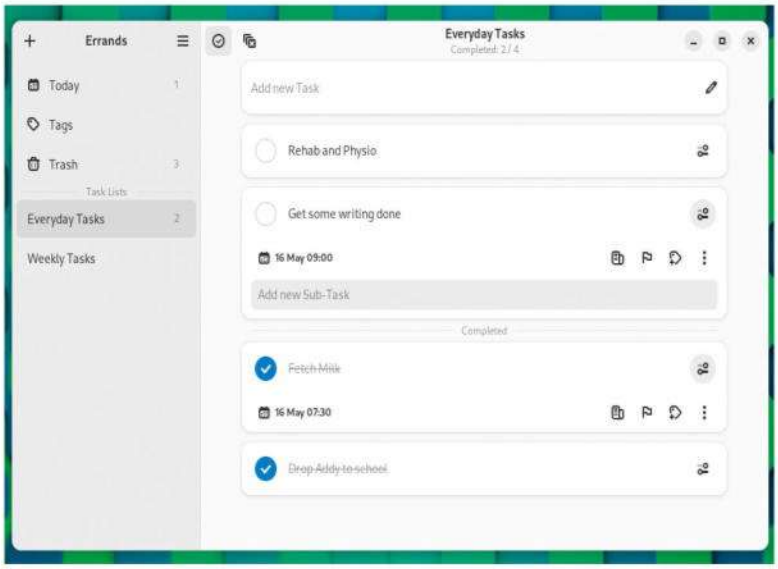
Version: 46.0.4 **Web:** https://github.com/mrvladus/Errands

There are loads of to-do list apps available, although most of the popular ones cater to teams of people working on bigger projects. While you can coax these apps to track day-to-day activities, most are overkill for such a task. If all you need to do is stay on top of daily chores, you need something a lot simpler, such as the aptly named *Errands*. You can use the app to ensure you complete your daily errands or, if you're a student, you can use it to organise your studies or homework.

The app is available on Flathub and can be installed with `flatpak install flathub io.github.mrvladus.List`.

On first launch, the app starts off empty, with the option to create a new list. Your list needs a name, such as Everyday Tasks. Inside the list, you can add as many tasks as you want. You can also add one or more subtasks to an existing task using the drop-down menu adjacent to that task.

You can also drag and drop the tasks to rearrange them. In fact, you can drag and drop tasks from one list to another as well.



Every task also has a key icon that toggles a toolbar. You can use this to set the start or due date and time for a task. You can also flesh out the task by adding detailed notes and assigning a priority. The toolbar also has a three-dot menu, which you can use to add accent colours to the task, edit it or delete it.

When you've completed a task, you can click on the radio button, which strikes through the task, while still keeping it in view. Or you can click the Delete Completed Tasks icon at the top of the interface to send completed tasks into the trash.

Errands can also sync your tasks with **Nextcloud Tasks**. Head to the app's **Preferences** to enable sync, and to hook it up to your **Nextcloud** instance.

FEDIVERSE CLIENT

Tuba

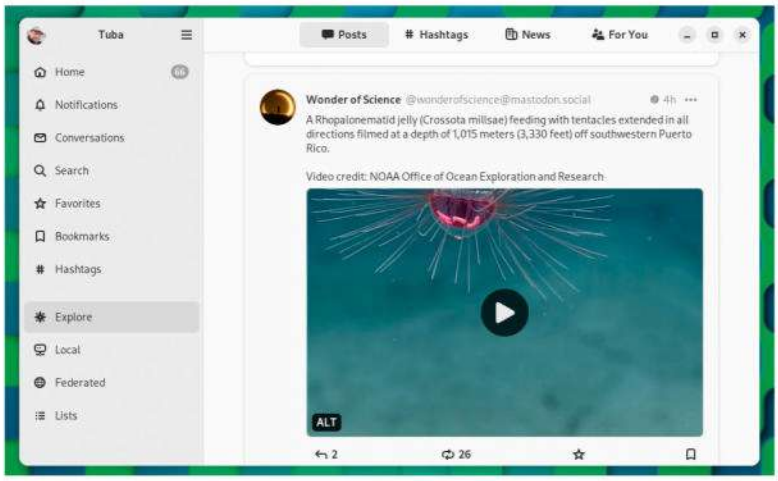
Version: 0.7.2
Web: https://tuba.geopjr.dev

While the fediverse has been around for a long time, Elon Musk's takeover of X-Twitter has people making a beeline for the decentralised federated social network. If you are one of those, you can use *Tuba* to stay on top of the goings-on in the fediverse. In other words, *Tuba* is for the fediverse what *Cawbird* was for Twitter.

Tuba is compatible with popular fediverse platforms including Mastodon, GoToSocial, Akkoma and others. The app is officially distributed on Flathub and you can install it with `flatpak install flathub dev.geopjr.Tuba`.

The first time you run *Tuba,* it asks you to enter the URL of your server. If you do not have a Mastodon account yet, the app directs you to a list of Mastodon servers. You can choose one and create your account, then go back to the app and enter the server URL.

This opens up a browser window and you are asked to log in and authorise *Tuba* to read and write to your new account. Once authorised, you are logged into your account in *Tuba*, and you can then go ahead and access the fediverse from there.



Use the sidebar in Tuba to access other streams, such as local and federated timelines, as well as viewing your bookmarked toots, and a lot more.

The app has an intuitive two-pane interface. Posts from accounts you follow appear under Home. Each post has buttons to reply, boost (same as retweet), favourite and bookmark a toot. Importantly, the app notifies you when a toot you've boosted has been edited to alter its content.

The Compose button is located in the bottom-right. It pops open the post editor, which has a character counter and support for media attachments. By default, posts are public, but you can mark posts as unlisted, followers-only or direct mentions, and even add a custom content warning.

The app also has a well-stocked Preferences dialog that you can use to fine-tune your *Tuba* experience.

PDF EDITOR

PDFsam

Version: 5.2.3 **Web:** https://pdfsam.org/pdfsam-basic

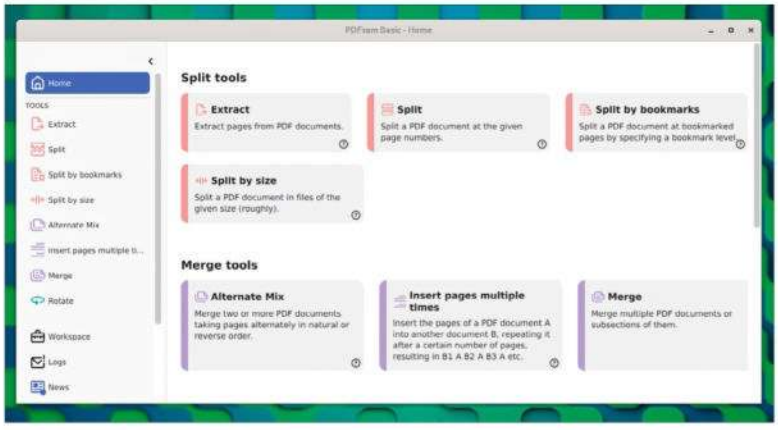
Anyone who regularly works with PDF files knows the value of a good PDF editor. *PDFsam Basic* offers some of the most commonly used functions for editing PDFs through an intuitive interface.

PDFsam Basic offers a DEB binary for Debian-based distributions, as well as a portable edition of the app in a compressed archive. If you're running a Debian-based distro such as Ubuntu, you can download the DEB file and install it with `sudo apt install pdfsam_5.2.3-1_amd64.deb`.

Users of all other distros, such as Fedora or OpenSUSE, need to grab the portable edition. After downloading, extract and run the app with:

```
$ tar zxvf pdfsam-5.2.3-linux.tar.gz
$ cd pdfsam-5.2.3/bin/
$ ./pdfsam.sh
```

The main window of *PDFsam Basic* lists all its supported features, divided into categories. They are also all listed in the left-hand column. Each function has a different interface. So while all the Split tools ask you for a PDF, their interfaces for splitting the source PDF vary.



In addition to multiple merge and split functions, you can also use PDFsam Basic to change the orientation of pages inside multiple PDFs at the same time.

For instance, you can use the Extract tool to extract every page, a range of pages, or specific pages. The Split PDF function looks very similar but offers the option to extract every even page or every odd page. Then there's the Split By Size function, which splits a PDF based on its size. This is useful for breaking up a bulky PDF (say 100MB), into chunks of smaller ones (such as 10MB each).

The Merge tools have several functions to combine multiple PDFs. For instance, the Alternate Mix function takes two or more PDFs and merges them into a single PDF, taking pages from each alternately. Of course, there's also a simple Merge function that melds multiple PDFs into a single PDF one after the other.

STRATEGY

Widelands

Version: 1.2
Web: www.widelands.org

It's perhaps the allure of bossing hordes of people at our whim that makes us love real-time strategy (RTS) games. While the genre isn't as popular as it once was, if you like RTS games, *Widelands* will be a fine addition to your collection. The game doesn't hide the fact that it's heavily influenced by *Settlers II,* but it offers a lot more.

The game is available in the official repos of most distros, and the developers also host a PPA for Ubuntu users. The best option though is to use the distro-agnostic Flatpak, which you can install with `flatpak install flathub org.widelands.Widelands`.

Widelands has both single-player campaigns and a multiplayer mode that enables you to play over the internet through the Widelands server, or with your friends over a LAN. It's best to begin with a single-player game. You can now choose to play a new game or a campaign. If you are new to the genre, or even just the game, it's a good idea to play through its tutorials to get to grips with the gameplay and the game's objectives.

When you start a new game, you get a list of maps. You can click on each to read about them. Each map also lists the size of the battleground, along with the total number of players it's designed for. You can now configure the gaming settings.

First, you have to select one of the game's five supported tribes. Each tribe has its own unique advantages and disadvantages in comparison to the others. Then you can define the win conditions. Hover over each of the seven win conditions to get a brief overview of what must be done to win the game. Finally, specify a playing time and you are all set to dive in.

The object of Widelands is to use the limited resources at your headquarters to expand your settlement by constructing military buildings and roads.

KART RACING

Dr Robotnik's Ring Racers

Version: 2.2
Web: www.kartkrew.org

Dr Robotnik's Ring Racers (DR3) is a *Mario Kart*-like kart racing game. It's created by the same team that produced the 3D *Sonic the Hedgehog* fan game *Sonic Robo Blast 2,* which we've featured in these pages (**LXF308**, *page 87). SRB2* was built using a modified version of the *Doom Legacy* port of *Doom.* If you are a fan of the *Sonic universe, DR3* will not disappoint.

The cross-platform game is available for Linux as a Flatpak that you can install with `flatpak install flathub org.kartkrew.RingRacers`.

Unlike the antagonist in the original, *DR3* reinvents Doctor Robotnik as the protagonist perched on a kart, collecting rings. *DR3's* gameplay is similar to any kart racer, seeing you drift through its 149 race tracks.

What sets *DR3* apart from other games in the genre is its focus on kart mechanics. The rings you collect are a form of currency that enable you to execute certain moves with your kart. There's an ebrake that can lock your car in place, a spin dash that works pretty much like it did in the original classic, trick pads that let you choose different types of boosts to activate, and lots more.

Understandably, the game has a tutorial to help you understand the mechanics of the kart the game, but *DR3's* stretched to almost an hour and couldn't be skipped. After a lot of brickbats, in the 2.1 release the developers added a handful of mechanisms to skip the tutorial, and get to the game proper.

The game has a Grand Prix mode with CPU opponents and multiple difficulty levels, and a Time Attack mode that allows replays. In multiplayer mode, you can enjoy *DR3* with friends in a four-player split-screen mode, or a 16-player online option.

Dr Robotnik's Ring Racers has all the popular characters from the Sonic universe including the iconic Sonic, Knuckles, Tails and over **50** other unlockable racers.

CREDIT: www.kartkrew.org, www.widelands.org

SCREENSHOTTER

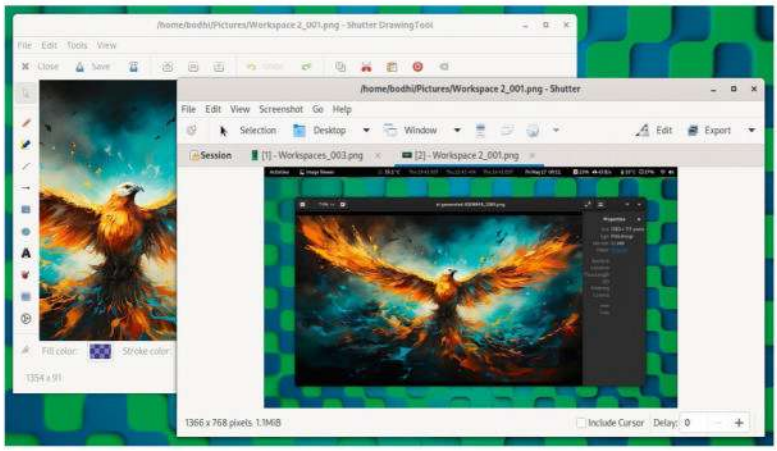
Shutter

Version: 0.99.5
Web: https://shutter-project.org

Virtually all distros ship with a basic tool that enables you to take screenshots of the entire screen, a specified section, or a particular window. Any more functionality is overkill for most people. However, if you find yourself using an external image editor to work on your screenshots, you can save yourself a lot of hassle and take screenshots with *Shutter* instead.

The app is available in the official repos of virtually all desktop distros. Ubuntu users can install it with `sudo apt install shutter`, while Fedora users can do a `sudo dnf install shutter`.

Shutter's interface looks rather busy for a screenshot app. To take a screenshot, head to File > New and choose the type of screenshot you want to capture. You can also use the button in the menu bar, which offers additional options. For instance, File > New > Desktop captures a screenshot of the current desktop. On the other hand, the Desktop menu button has a pull-down menu with which you can take a screenshot of another workspace, or all of them.



Shutter can also add several effects to your screenshots via plugins. Head to Screenshot > Run A Plugin to bring up a list of effects.

Once you have captured a screenshot, you can also use *Shutter* to edit it. Head to Screenshot > Edit to bring up *Shutter's* image editor. The editor has a vertical toolbar that provides various editing options.

To crop the screenshot to your desired selection or dimensions, select the last icon in the vertical toolbar. Then select a region or enter dimensions manually. You can emphasise and highlight a part of your screenshot using the highlights, shapes and arrows buttons. Select the one you want to use, then click-hold-drag over the specific region.

When done, you can save the screenshot in any of the popular image formats. You can also use *Shutter* to email the screenshot using your distro's default email app, or upload it to a local or remote folder.

REDACT DOCUMENTS

Obfuscate

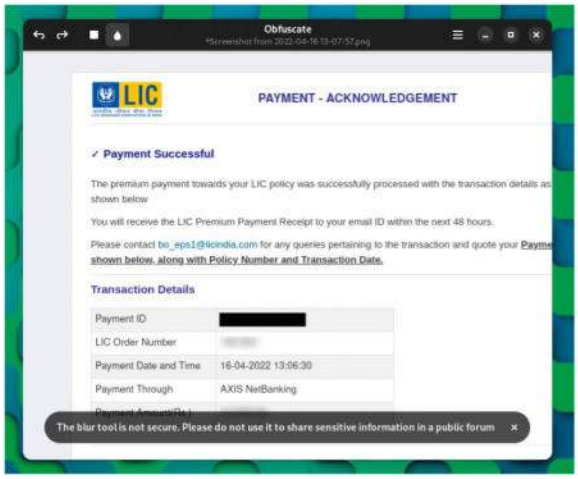
Version: 0.0.10 **Web:** https://gitlab.gnome.org/World/obfuscate/

If the screenshot you just took with *Shutter* reveals some information that you don't want anyone else to see, you can use the *Obfuscate* tool to conceal it.

Obfuscate is a straightforward app that justifies its name. The app offers two means to hide sensitive information from all kinds of images. *Obfuscate* is officially available on Flathub, and you can install it with `flatpak install flathub com.belmoussaoui.Obfuscate`. You can then fire it up from your distro's Applications menu or with `flatpak run com.belmoussaoui.Obfuscate`.

Obfuscate has a minimal interface. The app asks you for an image to do its magic. You can either drag and drop an image in the app's interface, or use the Open button to browse your filesystem. The app works with all the popular image formats, including JPEG, PNG, WEBP and more.

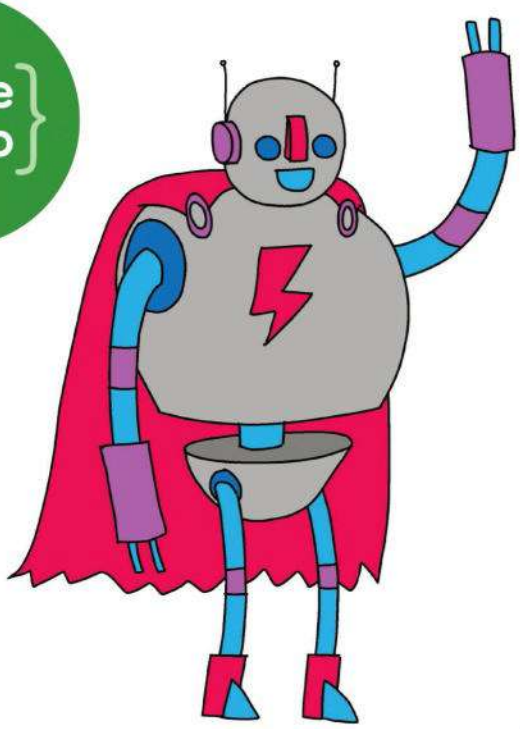
By default, the app loads the image at its native resolution. You can adjust the size of *Obfuscate's* window to match the resolution of the image. You can also reduce the size of the image. To do so, head to the hamburger menu and use the + and - buttons to increase or reduce the size of the image. You can also use the scroll bars to get to the portion of the image that you want to obfuscate.

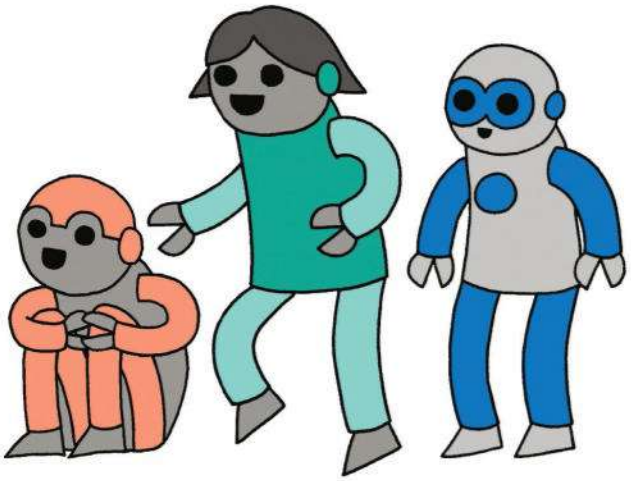


You can also use the blur tool to obfuscate the text you want to hide. But as the app warns, this isn't as secure as the fill tool.

When the bits you want to hide are in view, you can use the app to redact. *Obfuscate* offers two means to hide portions of an image. By default, the app uses the Fill tool, which is denoted by a square symbol in the app's toolbar. To use the tool, you simply draw over the text you want to redact. The app then fills the selected area with black. After obfuscating the bits you want to hide, head to the hamburger menu to save the redacted image. LXF

code club





Can you help inspire the next generation of coders?

Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at **www.codeclub.org.uk**

PLAY-BY-MAIL

Part Two! Don't miss next issue, subscribe on page 16!

Create a play-by-mail user interface

David Bolton looks further into designing play-by-mail-type games, with insights about the user interface and core game features.

OUR EXPERT

David Bolton worked as a game designer and programmer of play-by-mail games for the UK's biggest postal game company back before the web existed. He's working on new games now.

This is the second article in our series about creating play-by-mail games. These are games that are now played over the web (or by email), and we'll be looking at various aspects of this, including user input, use of JSON, clans, and free versus paid games.

Original play-by-mail (PBM) had one advantage compared to modern games: you didn't have to worry about user input. Game turns were entered on paper or specially printed cards, with an address on one side and a simple form with 10 lines on the other. You just filled in your game number and player number and orders, added a stamp and popped it in the letter box. We played *It's a Crime* back in 1986 and filled in orders on a card like as seen to the right

Now, though, there are two main ways to do it. First is emailing your orders. Just send in an email and it is processed manually or automatically. That actually takes more work, because someone has to copy the orders from email and get them into the game data. The other way is to create a website that not only lets you capture player input, but also enables you to exercise your ingenuity in displaying the game screens.

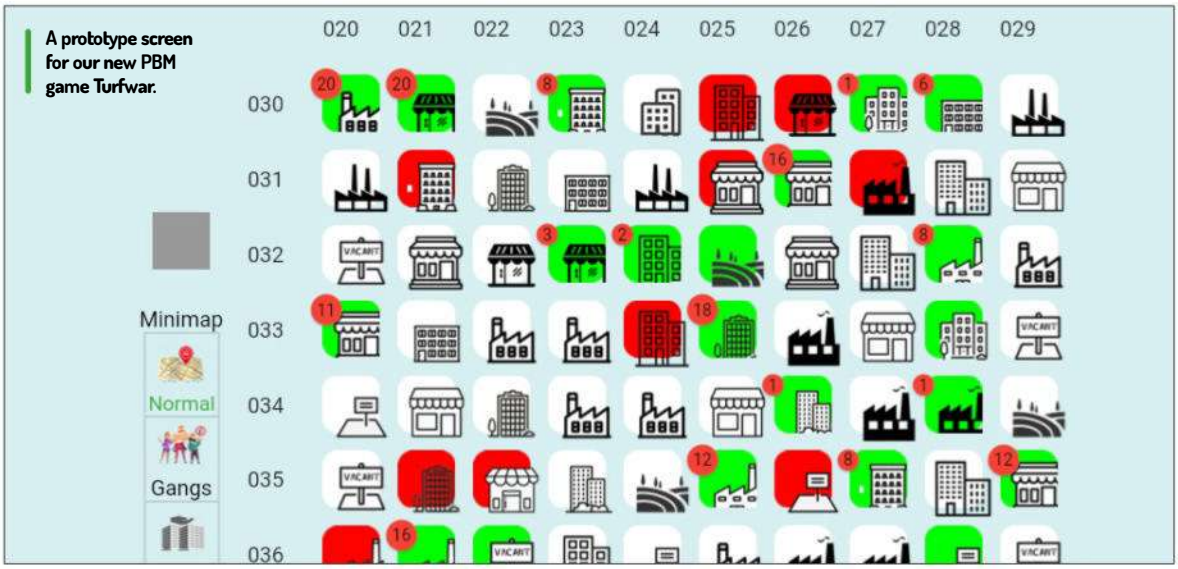
We'll look at the gangster game mentioned in the previous article – *Turfwar* – and consider how the user interface can be created.

Designing a user interface

The *Turfwar* game is based in a city with a map made up of city blocks, and these consist of four main types: residential (apartments), commercial (shops) and industrial (factories). There are also open blocks, such as parks, which aren't much use to players and provide no income.

If we have 100 gangs to start with and we give each gang two blocks as their starting point, and allow 10 unowned blocks per gang, that's a total of 1,200 blocks. Call it 1,225 and that gives a square shape of 35 x 35 blocks, or maybe a circular shape of radius 20, which is an area of 1,256 blocks.

That is too many blocks to show on screen at once – 8 x 8 or maybe 10 x 10 is probably the limit. We want to show other information as well. The prototype with the somewhat garish colours *(see screenshot, below)* shows 10 x 10 blocks. The grey rectangle on the left is a clickable mini-map, which while not currently



A prototype screen for our new PBM game Turfwar.

QUICK TIP

Try to get your game well play-tested. You can ask for players in places such as www.reddit.com/r/playmygame/. Dedicated play sessions with minutes between turns are the fastest way to do this.

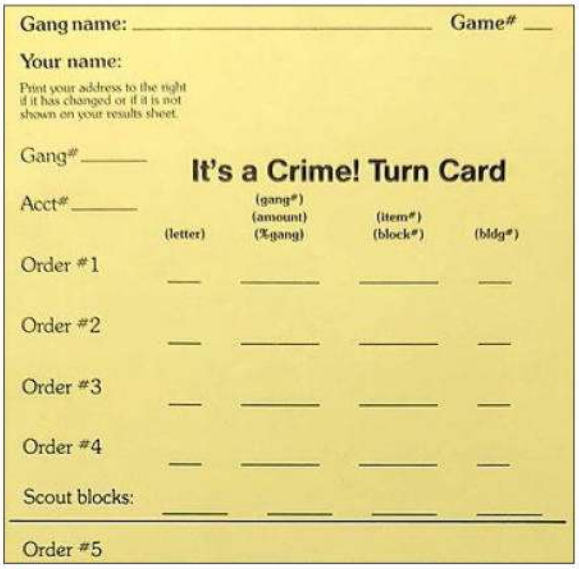
working, shows the whole 35 x 35 map. By clicking in it, you can move the 10 x 10 view to show anywhere in the city. Green areas are blocks that your gang owns, while red areas are blocks held by other gangs.

The four squares on the left are buttons, and clicking on them shows a different view, with the current view shown with green text on the button. The next screenshot *(over the page, top-left)* is a collage of the four different map views. The top-left shows just the gang-controlled areas with no details. Top-right is the default view, with a small circle showing how many soldiers are in each of your controlled blocks. Bottom-left shows details on each area you own – soldiers, cars, guns, with a dodgy font size – and the bottom-right one shows income from the businesses set up in the blocks you own. Having this interactive view is relatively easy in *Flutter* but would be harder in a normal web page. White blocks are the open ones or blocks that no one controls.

The graphics are not exactly great looking and a better way of displaying the map is underway, changing it to use Flutter Flame (**https://bit.ly/lxf316flame**) instead of the *Flutter* widgets that are currently used. Flutter Flame is a game engine that integrates with *Flutter* and makes it possible to have a draggable scrolling map. Flame supports tiled maps. These tiles can be created using the *Tiled* editor (**www.mapeditor.org**), which is an open source cross-platform editor for creating tiles.

It can be more interesting if there is a fog of war on game maps – this is a restriction that limits visibility. In *Turfwar,* the game map starts quite small at the beginning, perhaps only showing a 5 x 6 block view, with your two owned blocks and the empty blocks around them. With 100 player positions randomly squeezed into the 1,225 blocks, you are not going to be more than two or three blocks away from an enemy-owned block. But until you have spied on a block, you don't know whether it's uncontrolled or enemy held. As your territory expands and you spy on nearby blocks, the viewing area of the map expands and you start to see red blocks.

To speed this map expansion up and also add an element of strategic diplomacy if you meet another gang and become allies, you each get a copy of the other's map.


Gang name: _____ Game# ___

Your name:

Print your address to the right if it has changed or if it is not shown on your results sheet.

Gang#_____

Acct#_____

It's a Crime! Turn Card

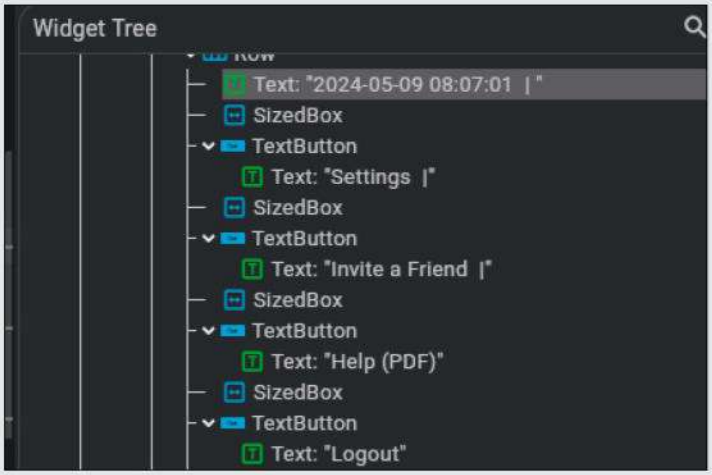
	(letter)	(gang#) (amount) (%gang)	(item#) (block#)	(bldg#)
Order #1	___	_____	_____	___
Order #2	___	_____	_____	___
Order #3	___	_____	_____	___
Order #4	___	_____	_____	___
Scout blocks:	_____	_____	_____	_____
Order #5				

An order card for the Its A Crime game from 1986.

In the previous article, it was explained how orders were done back in the original play-by-mail days. A command to recruit new soldiers might have the format R,5,20 where R is the command, 5 the number of soldiers and 20 the block we're recruiting from, and you'd write that on the turn postcard.

We don't use anything like that. Sure, internally we might send the order in that format, but we can make it easier for players to enter it by adding a pop-up menu when they click on a block. If it's a block owned by their gang, they get the commands such as Recruit, Create Business, Close Business and so on. If it's a block they don't own, the menu has commands such as Attack, Rob, Spy and so on. For Recruit, they get a pop-up where they enter the number of soldiers to send: 1-n, where n is the number presently unassigned in the block. Enter a number such as 5, press OK and that command is added to a list.

To make life easier for players, we also add a button that highlights blocks with unassigned soldiers. Each time you click it, it highlights one of your blocks. That way, you can work through all your owned blocks, giving orders. You can leave them there defending that block or give them orders. Unlike original PBM, you're not limited to 10 orders on the postcard, either. You



» WHY WE'RE ALL OF A FLUTTER

There's a number of ways to create a PBM user interface. One you should consider is programming it in *Flutter.* This is an SDK built on the Dart language that enables you to create beautiful-looking mobile apps for Android and iOS. It also lets you create Linux desktop apps and, most importantly, web apps. For more about *Flutter,* look up the article Creating Flutter Apps in issue **LXF306**.

The screenshot *(above)* comes from the prototype of the game *Kolonists*, a 100-player PBM set in an ocean full of small islands. Compared to a normal web page, *Flutter* web apps are typically redrawn up to 60 times per second. If there are lots of widgets (*Flutter* visual components) on screen being redrawn each time, this slows down the frame rate. But as it's a web page, this doesn't matter. Any values such as counts or date/times update without much work being done. This can give the game an air of real time.

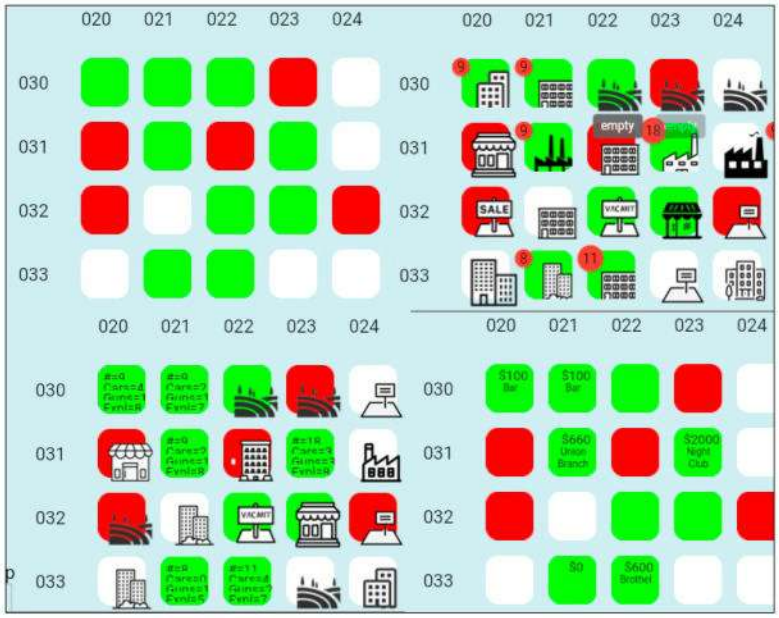
If you incorporate tooltips or badges, you can have tooltips appear for all important things.

(Above) Checking control layouts in Flutter in Android Studio on Ubuntu.

QUICK TIP

Game features that keep players in the game longer by, say, giving a player a bonus card after they've played so many turns can help to reduce churn. ***(See What Is Churn? section over page.)***

»



A collage of the four different map views.

give them orders until you've decided enough is enough. Now click the Submit Orders button (it's not shown on any of our screenshots) and the orders are uploaded to the website.

Whether you provide a way to view and edit/delete orders before you submit them or do a reset, clear all orders and re-enter them is up to you. Until it's time to process all orders, it should be possible to redo the orders and submit a new set that replaces any you've previously sent for this turn.

Handling orders and results

Modern programming languages are very good at handling JSON files. There's also XML, but it can be quite bulky in comparison to JSON. These files are used for holding orders and also the results. A further use is the game's data storage. In C#, if you declare classes with lots of public properties, it's possible to save the entire game data in a few JSON files.

PBM games are not too complicated in their data requirements. There's the game map, an array of player class objects, another one for gangs and one for game accounts. Many of these are held as a **List** of objects. By making each class inherit from a **Saveable<T>** class, it adds the capability of saving all public data for that class in one **Save** statement. The server can load all data for one game with this **Load** statement. The source for the **Savable** class is on GitHub (**https://bit.ly/lxf317save**) and an example of using it is in **https://learncgames.com/using-json-as-a-datastore-in-c/**

```
    public void Load()
    {
      try
      {
        city = new CityData();
        city.Blocks = (List<Block>)city.Load(Lib.
CityBlocksFilename)!;
        gangs = new AllGangs(city.Blocks);
        gangs.Gangs = (List<Gang>)gangs.Load(Lib.
GangsFilename)!;
        players = new Players();
        players.players = (List<Player>)players.
Load(Lib.PlayersFilename)!;
      }
      catch (Exception e)
      {
        // Report exception
      }
    }
```

The question is, why handle the data in this way, rather than using a database? Our thoughts on the main pros go like this:

1. It's simple and quick to save and load.
2. It can be backed up and restored just by file copying.
3. Fixing any issues can be done with a text editor.

The first two points would probably apply if you use *SQLite*. The JSON for a couple of blocks looks like this:

```
{
  "Id": 0,
  "X": 0,
  "Y": 0,
  "type": 1,
  "wealth": 1,
  "business": 0,
  "Influences": [],
  "Owner": -1,
  "Family": -1
},
{
  "Id": 1,
  "X": 1,
  "Y": 0,
  "type": 4,
  "wealth": 0,
  "business": 0,
  "Influences": [],
  "Owner": -1,
  "Family": -1
},
```

Other game features

Many PBM games support clans, alliances or federations. At its simplest, you can connect with other players in the same game and agree not to attack each other, at least for a while. Or a bunch of players agree to fight as one in their clan, for example. It's an attractive feature to include in your game and you have to code a way for players to connect within the game.

Connecting can be as simple as sending an in-game message. Bear in mind that some players may try to bully or intimidate other players through messages, so you must provide a way for players to block other players from contacting them. If players behave really badly, make sure there's a way to report them to the game's administrator. That's you.

Without going too much into it, you should draw up a set of rules for players that they have to agree to before they can join. The penalty for infringing the rules can be as much as permanently banning them from a game, or if they're really bad, permanently banning them from all games.

There are several ways to implement clans – for instance, let any two players who both agree form a clan. Others can join but you might want to build in a voting mechanism for clan members to approve or vote against later players joining the clan.

Make sure you set a maximum size for the clan, otherwise the first clan to get 50% of the game's

QUICK TIP

Make the game more interesting by adding things such as quests or targets. For instance, when your gang captures five blocks (or 10, and so on), they receive a bonus card. One example of a card might have the Feds raid (and close down) one of a rival gang's businesses.

players is going to dominate the game and be very hard to defeat. You might set it at 10% so in a 100-player game, no clan can exceed 10 players.

Once a clan has formed, you need to let them communicate within the game. It might be a web page where messages can be posted and seen by all clan players, or a way to send messages to all clan players. If you really want to enhance the game, allow a clan victory. After so many turns, all individual players must have joined a clan or be eliminated, and the clans fight until only one remains. Clans really do enhance the game and can help reduce churn.

What is churn?

If a game is free to join and play, it experiences higher rates of churn. This is a measure of how many players drop out over a game's life. When a player thinks they're not doing so well in a game, they drop out and join the next game that starts. Games where players have paid to join tend to have a lower churn rate; the players feel they have an investment in the game that makes them more likely to carry on.

You can discourage churn by penalising players who drop out – for example, by not allowing them to join another game for a week after they drop out. Or maybe give them a carrot after they've played in the game for so many turns; they receive a minor bonus in the game, such as a gift of extra weapons or vehicles in the *Turfwar* game, for instance, or a bonus card. It shouldn't be anything too big, though.

Free versus paid games

Monetisation was mentioned in the previous article. If you charge to play your game, it will probably take a long time to get players. Instead, consider letting anyone play for free, but if they pay a small annual fee to support the game, they receive enhancements to their game. Some ideas:

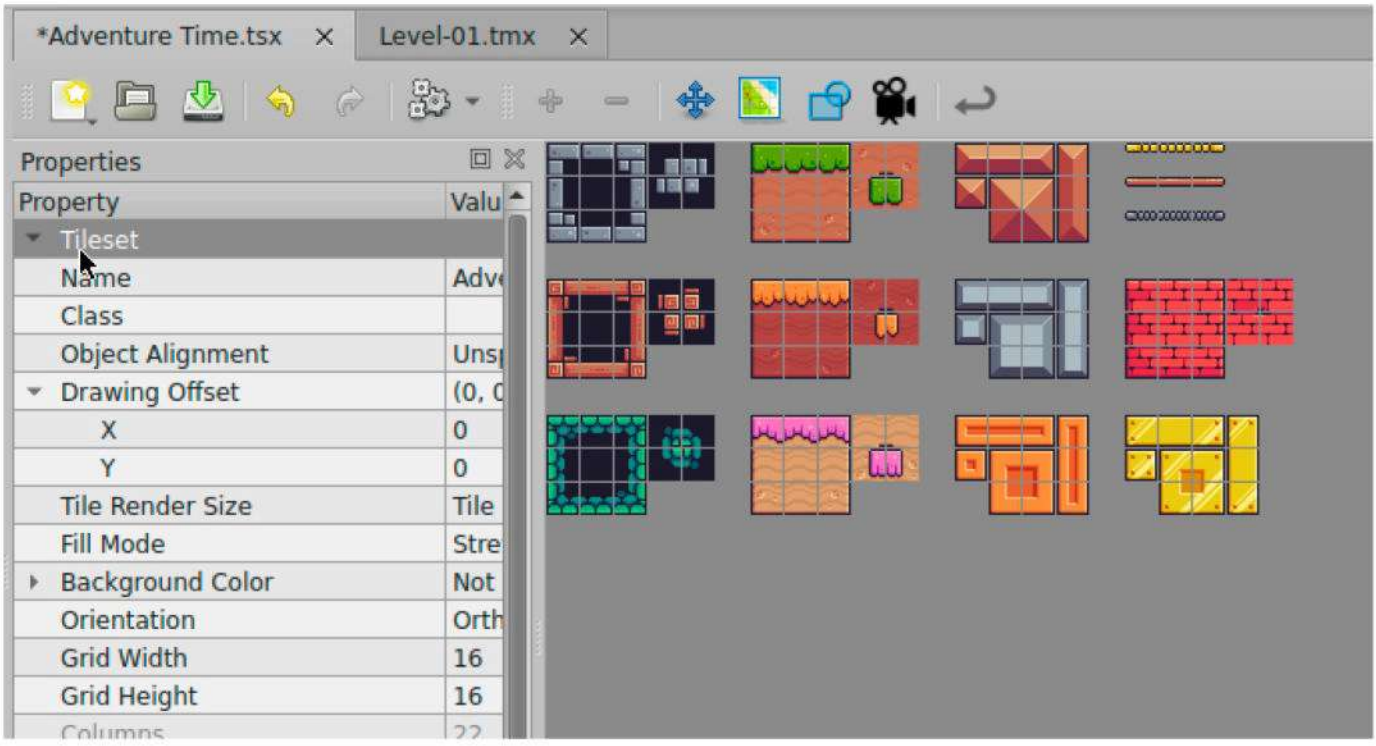
1. The ability to upload their own graphics for clans or player badges. These can be displayed on a game web page along with their text about them.
2. Add an API so some tech-savvy players can submit orders from their own software and get results directly. This is an advanced bit of code, so you need to make sure it's secure.
3. No adverts. It's likely that players will run ad-blockers but it is still possible to build adverts that aren't blocked into free play. The nicest way is to ask players to accept them but limit the ads to one per page and for them not to be in their face. LXF

» WHAT IS AN IN-GAME CURRENCY?

In theory, you could design a game and use real-world currency in it. But most people aren't economists or lawyers, and not aware of the problems it can cause. Using real money in a game opens you up to people trying to get hold of the money, and others, like the taxman, will raise an eyebrow and take a lot of persuading that you aren't doing naughty things like money laundering. It's a bad idea full stop.

Instead, it makes a lot more sense to use an in-game currency. For example, *Turfwar* uses dollars but they're just numbers. For instance, when you rob a block or get income from your businesses, it's in game-money dollars, not real dollars. You can't earn a real $4,000 from illegal activities in the game and withdraw it as real money.

To help support the game, you can encourage players to buy game merchandise or bonus cards (as mentioned in the previous article), and that's by spending real pounds and pence. Meanwhile, the gangs are accumulating dollars from their criminal activities and using it to buy guns, ammo and vehicles, and pay gangsters in the game. The game software keeps track of the money coming and going.



Tiled editor for creating tiles for Flame.

Part Three! Don't miss next issue, subscribe on page 16!

CLASSIC DEMOS

Screens of fractals and complex numbers

Blowing our minds and eyes with his psychedelic demos, it looks like **Ferenc Deák** has a rising medical bill to pay...

OUR EXPERT

Ferenc Deák never throws anything away, which is why he still has all his demoscene files.

QUICK TIP

You can find and clone all the code from the GitHub page: https://github.com/fritzone/lxf-demologia

Last month, we introduced some basic movement on to the screen, with the famous stretching-scrolling *Star Wars* effect, and with it also came the notion of the texture. Intentionally focusing on the educational aspect of the article, we have concocted a very suboptimal texture format, but one that is very easy to understand for beginners, too.

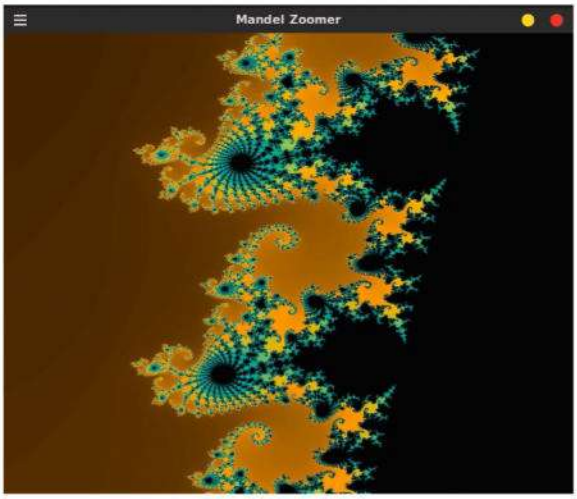
First we're introducing some wondrous concoctions known as fractals, and we will draw and zoom them around the screen. For the second part of this month's tutorial, we are focusing on some stunning effects in order to demonstrate how clever texture manipulations can create some unexpected results. One of them is the rotozoom effect (an effect that rotates and zooms in and out) and the next one is the famous tunnel effect, which gives the viewer the illusion of flying through a tunnel.

But before we delve into more details for these effects, we need to lay the mathematical foundation used in implementing them. So dear reader, please head over to our boxouts and read upon a bit on the dark art of trigonometry and the mysterious concoction of complex numbers. Because we will use both of those notions heavily in this episode.

Biting our own tail

Fractals are intricate and self-replicating geometric patterns characterised by self-similarity, where parts of the structure resemble the whole at different scales. Generated through iterative processes or mathematical equations, fractals exhibit infinite detail and complexity emerging from a simple formula. They are found in nature, from clouds and mountains to biological structures, such as the pattern found on trees or ferns *(Polypodiopsida,* for lovers of Latin names). Fractals play a prominent role in computer graphics, art and scientific simulations, with examples such as the Mandelbrot set and the Julia set showcasing their visually captivating and mathematically intriguing properties.

The beauty of the Mandelbrot set lies in its intricate and self-replicating patterns, which can be explored at various levels of magnification limited only by the capabilities of the hardware. There are several sections



The seahorse valley in the Mandelscape.

that you can discover in the set, after zooming into various areas:

- **Mini Mandelbrots** (Mandelbabies): At various zoom levels, you can find smaller replicas of the entire Mandelbrot set within the set itself, and at deeper zoom levels, these babies are rotated with various angles and have various degrees of distortion, mostly caused by the limited capacity of computers to represent very small numbers.
- **Seahorse valley** This region, found near the main cardioid of the set, contains intricate seahorse-like shapes. (A cardioid is a mathematical curve that resembles a heart shape.)
- **Satellite structures** Small fractal-like satellites or bulbs surrounding the main cardioid and connected by thin filaments.
- **Julia sets** Named after French mathematician Gaston Julia, the Julia fractal can be found embedded deep in the Mandelbrot set at various locations.

Drawing Mandelbrot sets is not very complicated, due to the simplicity of the formula: `z_{n+1} = z_n^2 + c`.

Here, `z_n` is a complex number representing the state of iteration at step `n`, and `c` is a constant complex number that represents the initial point being tested in the complex plane. The iteration continues

until the current `|z_n|` (the magnitude of `z_n`) becomes larger than a specified threshold, or until a maximum number of iterations is reached, and the value of `n` is the colour used to plot the current fractal point. The code doing this can be quickly written like this:

```
void updateScreen(Uint8* screen, double zoom, double
centreX, double centreY) {
        for (int x = 0; x < SCREENSIZE_X; x++) {
        for (int y = 0; y < SCREENSIZE_Y; y++) {
        double zx = (x-SCREENSIZE_X/2)/
(zoom*SCREENSIZE_X)+centreX;
        double zy = (y-SCREENSIZE_Y/2)/
(zoom*SCREENSIZE_Y)+centreY;
        double cx = zx, cy = zy, zx2 = zx*zx, zy2=zy*zy;
        Uint8 colour = 0;
        while (zx2 + zy2 < 4.0 && colour < 256) {
        zy = 2.0 * zx * zy + cy;
        zx = zx2 - zy2 + cx;
        zx2 = zx * zx;
        zy2 = zy * zy;
        colour++;
        }
        putPixel(x, y, colour, screen);}}
}
```

The following variables are important while drawing the fractal:

(centreX, centreY) is the centre of the fractal. We have used **(-0.743023954, -0.129123012)** as being the centre of the seahorse valley.

zoom is the current level of zoom. The bigger the level, the deeper we zoom in the fractal around the centre of it. Zoom level 1 means no zoom, while 512 is a really deep zoom.

What the function does, besides following the algorithm presented above, is correctly calculate the bounds of the fractal in relation to the size of the screen and the zoom factor, then apply the formula in the inner loop and simply draw it on the screen.

One of the most interesting effects that can be achieved with fractals is the fractal zoomer. This, as the name suggests, zooms into the depths of the fractal, usually on a linear scale, thus revealing in real time the beautiful details of the Mandelbrot set.

In order to achieve this, we need to cleverly manipulate the zoom level and – why not? – the centre of the fractal, too, to achieve the desired effect. Writing this code is not that complicated – it should be along the lines of simply increasing a double parameter that represents the zoom level passed in to the function, and we leave it as an exercise for you (or you can check it out on the project's GitHub site).

Certainly, there is more than one set of fractals on the planet, but the Mandelbrot set is easily the most famous, so we have focused our code around it. There are more optimised methods of drawing the set, so we encourage anyone interested in this domain to boldly go where all the knowledge of the world is, and grab a book or read another article about these wondrous creations of the mind.

» ENTER THE TRIANGULUM

Basic trigonometric functions are defined in relation to the angles of a right-angled triangle. Consider a right-angled triangle, where one angle is a right angle: 90°. The triangle's sides are labelled as follows:

- **Hypotenuse (h)** The side opposite the right angle.
- **Adjacent (a)** The side adjacent to the angle of interest but not the hypotenuse.
- **Opposite (o)** The side opposite the angle of interest but not the hypotenuse.

We also need to define the unit circle, which is just a simple circle with a radius of one unit centred at the origin in the Cartesian plane. Points on this circle, denoted by coordinates (x,y), represent angles measured anticlockwise from the positive x-axis. As the angle increases from 0 to 2π radians (or 0° to 360°), points on the unit circle trace out sinusoidal curves, visualising the periodic nature of these trigonometric functions:

Sine

$\sin\theta = o/h$, or the sine of an angle is the ratio of the length of the side opposite the angle (o) to the length of the hypotenuse (h). In a unit circle, the y coordinate of a point on the circumference corresponding to an angle θ is the sine of that angle.

Cosine

$\cos\theta = a/h$, or the cosine of an angle is the ratio of the length of the side adjacent to the angle (a) to the length of the hypotenuse (h). In a unit circle, the x coordinate of a point on the circumference corresponding to an angle θ is the cosine of that angle.

Tangent and arctangent

$\tan\theta = o/a$, or tangent is the ratio of the length of the side opposite (o) the angle to the length of the adjacent (a) side. In mathematical terms, if $\tan\theta = x$ then $\text{atan}(x) = \theta$, so the arctangent is the inverse of the tangent. The arctangent function is often used to find the angle in a right-angled triangle when the lengths of two sides are known.

You spin me right round...

Before continuing, if you haven't already read the boxout concerning trigonometry *(above)*, now is the time to do so... And welcome back. Now that we have armed ourselves with some basic trigonometric

Pirouetting punk penguin peers peekingly, people-peeping.

ENTER THE IMAGINARIUM

Before we continue our adventure in demo land, just a quick recapitulation of our most beloved friends, numbers. They can be real numbers, R, and imaginary numbers, C. Real numbers mostly fit in the following three major categories:

- **Rational numbers** Numbers expressible as the quotient or fraction of two integers, where the numerator (the number above the fractional line) is an integer, and the denominator (the number below) is a non-zero integer. Examples include fractions like ½, -¾.
- **Irrational numbers** Numbers that cannot be expressed as fractions of two integers, with non-terminating, non-repeating decimal representations. Examples include √2 or π.
- **Integers** Whole numbers, positive or negative, including zero.

Real numbers form the foundation of mathematics and are essential for representing quantities in the real world, and serve as a basis for more advanced mathematical structures.

Complex numbers are mathematical entities that extend the concept of real numbers to include a new component, known as the imaginary unit, denoted by `i`. The imaginary unit is defined as `i^2 = -1`. A complex number is expressed in the form `a + bi`, where `a` and `b` are real numbers, and `i` is the imaginary unit.

- **Real part** (`a`): Represents the conventional real number part of the complex number. It is positioned on the horizontal axis of the complex plane.
- **Imaginary part** (`b • i`): Involves the imaginary unit `i` multiplied by a real number `b`. It is positioned on the vertical axis of the complex plane.

While real numbers are only one-dimensional entities, meaning they can be represented on a line, the complex plane is a two-dimensional coordinate system, where the horizontal axis represents the real part, and the vertical axis represents the imaginary part. Complex numbers can also be represented in polar form (`r • (cos(θ) + i • sin(θ))`), where `r` is the magnitude and `θ` is the argument of the complex number. (Please read the other boxout *(previous page)* for an introduction to `sin` and `cos`.)

We are using complex numbers to introduce one of the most visually stunning complex mathematical structures that any self-respecting demo of the early '90s couldn't be imagined without: the famous Mandelbrot fractal.

knowledge, we should be ready to dig into slightly more advanced formulae.

In order to implement the rotozoom effect, we need to, well, rotate something, which in our case is a point (and another one, and another, until they add up to a texture), so the first new formula we learn is the rotation formula of a point around another point. Let's say you have a point `P(x, y)` that you want to rotate around another point `O(a, b)` by an angle `θ` in an anticlockwise direction. The coordinates `(x', y')` of the rotated point `P'` can be calculated using the following formula:

```
x'=(x–a)•cos(θ)–(y–b)•sin(θ)+a\\
y'=(x–a)•sin(θ)+(y–b)•cos(θ)+b
```

In order to keep our life simple, we just assume that we want to rotate around the origin – the point `O(0,0)` – then we just can entirely ignore the `a` and `b` values and our formula is reduced to something digestible. This surprisingly short code snippet is the full implementation of the rotozoom:

```
void updateScreen(Uint8* screen, const
std::vector<int>& imageData) {
        angle = (angle + SPEED ) % 360;
        auto rad_angle = angle * M_PI / 180.0;
        auto sin_angle = sin(rad_angle);
        auto cos_angle = cos(rad_angle);
        auto zoom_factor = cos_angle * 1.1;
        for (int x = 0; x < SCREENSIZE_X; x++) {
        for (int y = 0; y < SCREENSIZE_Y; y++) {
        int u = static_cast<int>((x * cos_angle - y * sin_
angle) * zoom_factor) % TEXTURE_SIZE_X;
        int v = static_cast<int>((x * sin_angle + y * cos_
angle) * zoom_factor) % TEXTURE_SIZE_Y;
        while(u < 0) u += TEXTURE_SIZE_X;
        while(v < 0) v += TEXTURE_SIZE_Y;
        auto pixel = static_cast<Uint8>(imageData[ u *
TEXTURE_SIZE_X + v]);
        putPixel(x, y, pixel, screen);
        }
        }
}
```

The method takes in the **imageData** parameter, representing the texture in the usual flat buffer format. The following calculations update the current rotation angle (which is represented using degrees) with a magic variable called **SPEED**, which, as the name suggests, is the speed of the rotation. We don't want our poor punk penguin to suffer from motion sickness, so we just use **1** for this value. The `% 360` ensures that the rotation angle is kept in the sane values interval, but as for the purpose of 360 itself, you will have to ask the ancient Babylonians. Then the **rad_angle** value is calculated, being the representation in radians (radians are used in the scientific/programming community) of the current degrees. We could have skipped the conversion step altogether, and kept the angle in radians, but it would not have been so self-explanatory.

After this, we calculate the sine and cosine of the current angle, and also the current zoom factor. Back in the day, it was wise to store a precalculated sine and cosine table, but as processor speed is quite decent nowadays, we just use the functions as they are.

By using a periodically changing zoom factor, such as the value of the cosine, we make sure that the poor punk penguin is also oscillating and zooming while rotating around. The secret ingredient is hiding in the following lines:

```
int u = static_cast<int>((x * cos_angle - y * sin_angle) *
zoom_factor) % TEXTURE_SIZE_X;
int v = static_cast<int>((x * sin_angle + y * cos_angle) *
zoom_factor) % TEXTURE_SIZE_Y;
```

You can easily recognise the formula for rotation around the origin we have presented a few lines above. In graphics programming, the pair `(u,v)` is generally used for texture coordinates, called texels (a nice combination of *tex*ture and pix*el*) and used to represent the coordinates of the pixel in a given texture. The formulae above do nothing else, but calculate a corresponding texel `(u,v)` for the current `(x,y)` pixel coordinates, taking into consideration the current **zoom_factor**, and making sure that the value is constrained in the required interval of the size of the texture, by using the **% TEXTURE_SIZE_*** operation.

Then, using the formula presented in the first instalment of this series, we find the colour of the pixel

in the texture at the given texel coordinates and just plot it on the screen at the current `(x,y)` coordinates.

And that's it. Easy, peasy, whatvereasy. After successfully compiling and linking it, if you have used our code and texture from the GitHub, you'll see the final scene appear *(see screenshot, page 95)*.

Light at the end of...

The final effect we've reserved for this month is the famous tunnel effect. It is similar to the rotozoom in the way that it has a backbone of manipulating the pixels of a texture in a way to present it as a tunnel, but it is one step more complicated. The principle is the same, but the formula for choosing a texel is different.

We have used our tool from last month and created a 256x256 texture for this effect. However, that should not stop you from using your own textures and generate what you consider to be fun.

The tunnel's centre is represented by the variables **TUNNEL_CENTRE_X** and **TUNNEL_CENTRE_Y**, which are initialised to the centre of the screen:

```
static const int TUNNEL_CENTRE_X = SCREENSIZE_X / 2;
static const int TUNNEL_CENTRE_Y = SCREENSIZE_Y / 2;
```

The idea behind the method for choosing the pixel for the tunnel effect can be presented in the following lines of code:

```
int distance = (DISTORTION * TEXTURE_SIZE /
sqrt(pow(x - TUNNEL_CENTRE_X, 2) + pow(y -
TUNNEL_CENTRE_Y, 2)) );
int angle = (MULTIPLICATOR * TEXTURE_SIZE *
atan2(x - TUNNEL_CENTRE_X, y - TUNNEL_
CENTRE_Y) / M_PI );
int u = (distance + TEXTURE_SIZE * animation_zoom
) % TEXTURE_SIZE;
int v = (angle        + TEXTURE_SIZE * animation_
rotation) % TEXTURE_SIZE;
```

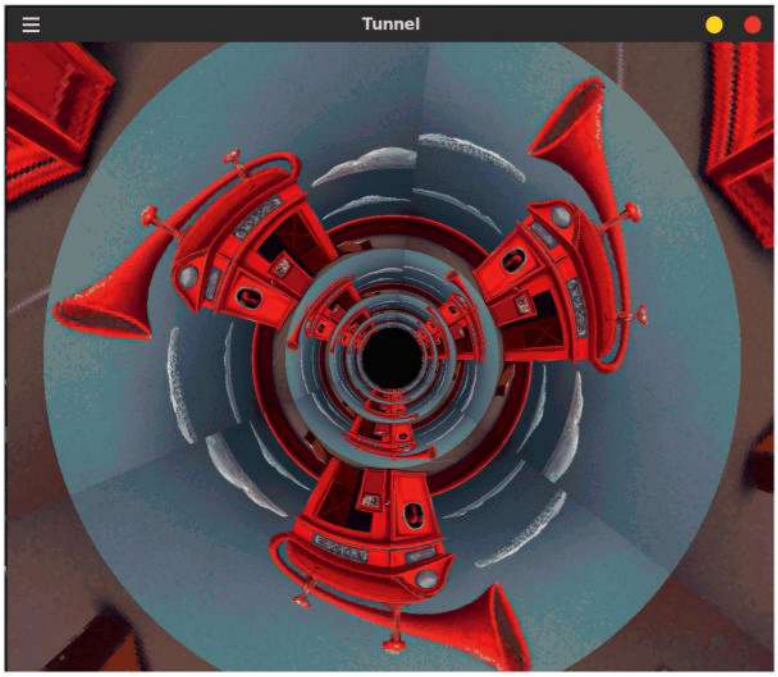
The variable **TEXTURE_SIZE**, as its name suggests, is the size of the texture. In order for this effect to work as effectively as possible, we need square textures, whose size is a power of two. The variable **DISTORTION** is used to manipulate how dense the texture will be – it is recommended to be a multiple of the size of the texture.

Going into more detail, the code does the following: firstly, for each pixel, while drawing it on the screen, we calculate a value being the inverse of its distance from the centre of the tunnel multiplied by the size of the texture and the required distortion, using the formula:

```
\begin{equation}
\text{{distance}} = { \text{{D}} \times \text{{T}}} \times \
frac{1}
        {{\sqrt{{(x - \text{{TX}})^2 + (y - \
text{{TY}})^2}}}}
\end{equation}
```

Here, `D` is the **DISTORTION**, `T` is **TEXTURE_SIZE**, `TX` is **TUNNEL_CENTRE_X** and `TY` is **TUNNEL_CENTRE_Y** – we just shortened them in the formula. This gives the illusion that the closer we get to the centre of the tunnel, the further away the pixels are.

There is also the possibility to not to use the square root, but simply calculate `{{(x - \text{{TX}})^2 + (y - \text{{TY}})^2}}` divided by large number, a multiple of the texture size, but in our experience, that gives a slightly distorted texture at the edges, while using the logarithmic function instead of the distance calculation gives a much slower motion.



There is **no light at the end of the tunnel, but we can easily program in a light bulb.**

The **angle** value is calculated as being the angle of the current pixel when participating in the construction of an angle with the centre of the tunnel, and this will wrap around when it reaches a full circle, giving the required circularity for picking the texels from the texture

The texel calculation formula consists of just picking the already calculated distance value for the **u** coordinate, the calculated angle value for the current pixel as the **v**, and making it wrap around in the bounds of the texture. When correctly applying the **animation_zoom** and **animation_rotation** values, we can finally put all in place, and the final screen *(see above)* awaits us after compiling and running the application.

Please note that the black hole at the end of the tunnel is the place where the light should be, but in reality, a very ugly circular moiré pattern would have evolved there due to the compression of the texture. In order to not to show the ugliness, for now we just did a quick check of whether the current pixel is inside a circle or not, and coloured the pixel accordingly. But it is up to you whether you want to fade out the colours or – why not? – add the famous light bulb in there.

The future, behold

Picking texels, based on various formulae and algorithms, is the basis of a lot of effects that we were unable to present here due to lack of space, but these effects always give very visually satisfying results, too, so we really encourage you to experiment with the code above, mix in some more magic, and wonder what comes out. Or just wait until next month, when we will do some more experiments and present some more mind-bending effects. Happy coding! LXF

QUICK TIP

The book *The Fractal Geometry of Nature* by none other than Benoît Mandelbrot details a lot of the wonders of these miraculous constructs, and is recommended reading for anyone interested in the nature of complex, beautiful, self-replicating structures, called fractals.

NEXT MONTH

LXF318 will be on sale Tuesday 23rd July 2024

CREDIT: Magictorch

VPN PACKET PROTECTION!

We trace our network packets from source to destination to discover how to secure our VPNs.

Get chatty with open source

It's good to talk, as an old advertising slogan used to say, but how? We test the best secure distributed chat tools.

Become a podcasting pro

The tools, tips and tech you need to create a smash-hit podcast, from the perfect home studio to better editing.

Save your old documents!

Discover how *LibreOffice* supports your old Microsoft documents better than Microsoft can…

Custom routers

Build and configure your own wireless router – grab advanced features, VPN and total control.

Content of future issues subject to change. Our packets might still be stuck in Russia…

LINUX FORMAT
The #1 open source mag

Future Publishing Limited, Quay House, The Ambury, Bath, BA1 1UA
Email contact@linuxformat.com

EDITORIAL
Editor-in-chief Neil Mohr
Art editor Fraser McDermott
Production editor Katharine Davies
Group editor-in-chief Graham Barlow
Group art director Warren Brown
Editorial contributors
Paul Alcorn, Mike Bedford, Denise Bertacchi, Jonni Bidwell, David Bolton, Neil Bothwick, Stuart Burns, Ferenc Deák, Nate Drake, Matt Holder, Jon Masters, Dave Meikleham, Nick Peers, Les Pounder, Michael Reed, Mayank Sharma, Shashank Sharma, Alex Wawro, Andy Williams
Cover illustration Magictorch.com

Content production Adequate Media Limited

ADVERTISING
Commercial sales director Clare Dove
clare.dove@futurenet.com
Advertising director Lara Jaggon
lara.jaggon@futurenet.com
Account director Andrew Tilbury
andrew.tilbury@futurenet.com

INTERNATIONAL LICENSING
Head of print licensing Rachel Shaw
Linux Format is available for licensing and syndication.
To find our more contact us at licensing@futurenet.com
or view our content at www.futurecontenthub.com

NEW SUBSCRIPTIONS & BACK ISSUES
Web www.magazinesdirect.com
UK 0330 333 1113 **World** +44 (0) 330 333 1113

EXISTING SUBSCRIPTIONS
Web www.mymagazine.co.uk
UK 0330 333 4333 **World** +44 (0) 330 333 4333
Subscription delays: Please allow up to seven days before contacting us about a late delivery to **help@magazinesdirect.com**

CIRCULATION
Newstrade & retail category director Ben Oakden

PRODUCTION AND DISTRIBUTION
Group head of production Mark Constance
Production manager Nola Cokely
Senior ad production manager Jo Crosby
Digital editions manager Jason Hudson

THE MANAGEMENT
Managing director technology group Paul Newman
Global head of design Rodney Dive
Commercial finance director Tania Brunning
Printed by William Gibbons & Sons
Distributed by Marketforce UK
121-141 Westbourne Terrace, London W2 6JR. **www.marketforce.co.uk**
For enquiries email: **mfcommunications@futurenet.com**

Order and access back issues: If you are an active subscriber, you have instant access to back issues through your iOS or Android device/s. Your digital magazine entitlement is available at no additional cost and no further action is required. Pocketmags library may not have access to the full archive of digital back issues. You will only be able to access the digital back issues as long as you are an active subscriber.

To purchase single back issues (print format only): Visit www.magazinesdirect.com (click on Single Issues tab) or email help@magazinesdirect.com. For further help call +44(0)330 333 1113. Lines are open Mon- Fri 8.30am-7pm and Sat 10am- 3pm UK time. Magazinesdirect.com is owned and operated by Future Publishing Limited.

BAR rates for Linux Format: £84.37 for UK, €171 for Europe, $194 for USA, £149 for rest of world.

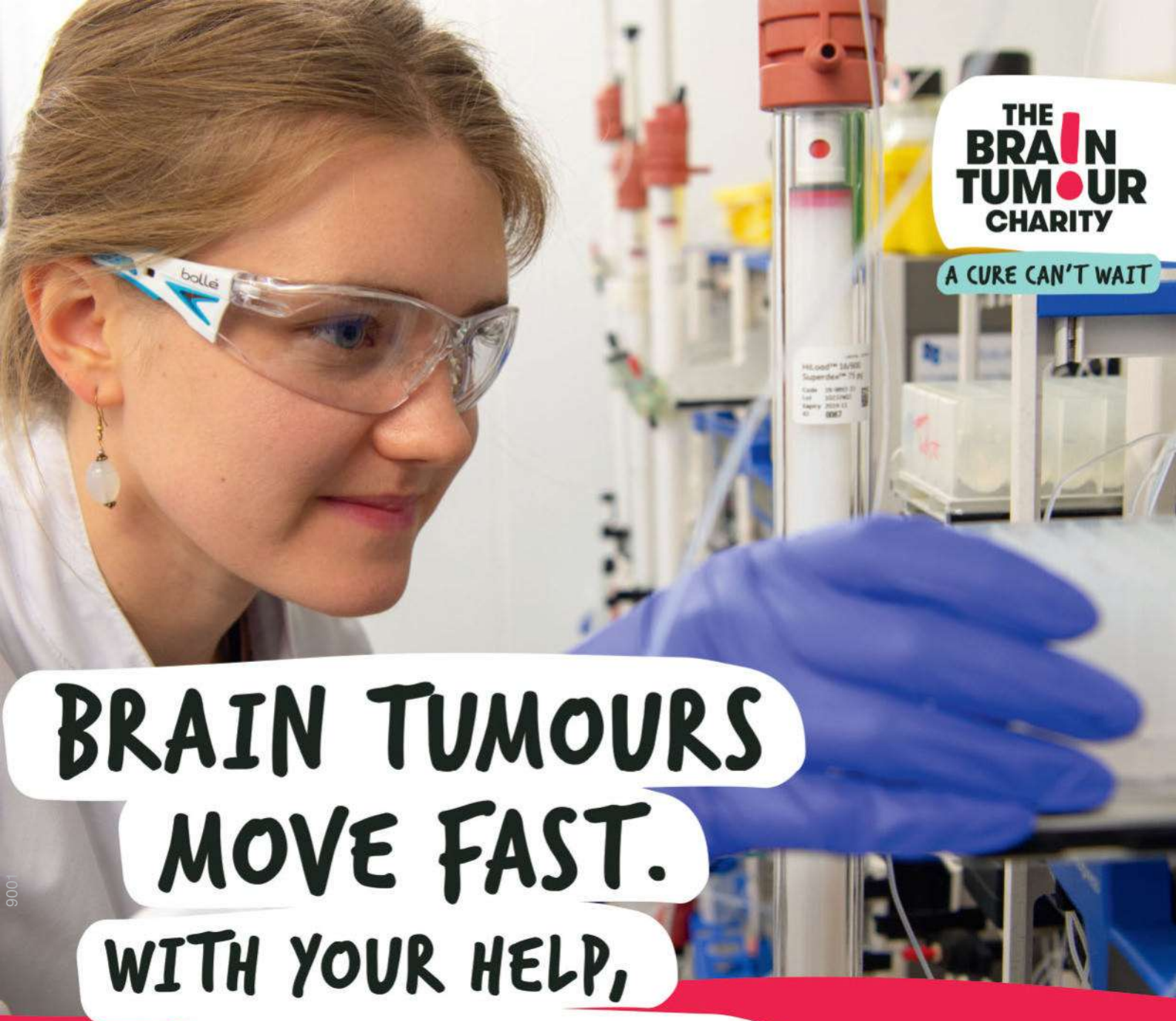
Linux® is the registered trademark of Linus Torvalds in the US and other countries. GNU/Linux is abbreviated to Linux throughout for brevity. Where applicable, code printed in this magazine is licensed under the GNU GPL v2 or later. See www.gnu.org/copyleft/gpl.html.

The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

EFF.ORG

ELECTRONIC FRONTIER FOUNDATION

Protecting Rights and Promoting Freedom on the Electronic Frontier



We're working to create a future where brain tumours are curable. We urgently need your help to accelerate research.

Text DEFEAT5 to 70507 to donate £5, please help us to find a cure.

thebraintumourcharity.org